FOURIER ANALYSIS OF ARBITER PHYSICAL UNCLONABLE FUNCTIONS

BENJAMIN ZENGIN

Influence and PAC Learnability of Arbiter PUFs

Master Thesis

Institut für Informatik
Fachbereich Mathematik und Informatik
Freie Universität Berlin
24th July 2017

Benjamin Zengin: *Fourier Analysis of Arbiter Physical Unclonable Functions*, © 24th July 2017

SUPERVISORS:
Prof. Dr. Marian Margraf

LOCATION:
Berlin

TIME FRAME:
24th July 2017

I want to thank everyone who supported me during the making of this thesis.

Special thanks go to Nils Wisiol, Tudor Soroceanu, and Marian Margraf.

I dedicate this to the most important persons in my life.

E. and M.

ABSTRACT

Physical Unclonable Functions (PUFs) have emerged as a promising alternative to conventional mobile secure authentication devices. Instead of storing a cryptographic key in non-volatile memory, they provide a device specific challenge-response behavior uniquely determined by manufacturing variations. The Arbiter PUF has become a popular PUF representative due to its lightweight design and potentially large challenge space. Unfortunately, opposed to the PUFs definition, an Arbiter PUF can be cloned using machine learning attacks. Since PUFs can be described as Boolean functions, this thesis applies the concept of the Fourier expansion on Arbiter PUFs, focusing on the notion of influence. Based on this, statements about its Probably Approximately Correct learnability are deduced and discussed.

ZUSAMMENFASSUNG

Physical Unclonable Functions (PUFs) haben sich zu einer vielversprechenden Alternative zu konventionellen, mobilen Authentifizierungsgeräten entwickelt. Anstatt eines in nichtflüchtigem Speicher abgelegten kryptografischen Schlüssels, bieten sie ein gerätespezifisches challenge-response Verhalten, das einzigartig durch Fertigungsungenauigkeiten bestimmt wird. Die Arbiter PUF ist aufgrund ihres leichtgewichtigen Designs und potentiell großen Challengeraums zu einem bekannten PUF Repräsentanten geworden. Bedauerlicherweise, entgegen der Definition von PUFs, kann die Arbiter PUF mithilfe von Machine Learning Angriffen geklont werden. Da PUFs als Boolesche Funktionen beschrieben werden können, wendet die vorliegende Arbeit das Konzept der Fourier Erweiterung auf die Arbiter PUF mit Fokus auf den Begriff des Influences an. Darauf basierend werden Aussagen über ihre Probably Approximately Correct Erlernbarkeit gefolgert und diskutiert.

CONTENTS

LIST OF FIGURES

1 INTRODUCTION

Since the beginning of mankind, information was one of the most valuable goods. The immediate consequence is the need to protect it. The ability to share information secretly and exclusively between two or more parties developed throughout the history and established the art of cryptography. The techniques used in this process evolved from the basic Caesar cipher to modern cryptographic methods such as the Advanced Encryption Standard (AES). Beginning with the era of computers more complex mathematical constructions emerged as a new method to model cryptographic systems. Nowadays we classify a cryptographic system by four major goals.

1. Confidentiality: Only the authorized parties can read the contents of a transmitted message.

2. Authenticity: The identity of the author of a message can be verified.

3. Integrity: No unauthorized party is able to modify the contents of a message.

4. Non-repudiation: The creator of a message must not be able to deny the authorship.

A cryptographic system, however, does not have to satisfy all of these goals. For instance, a symmetric cryptographic system such as AES satisfies only the first three goals. On the other hand, a signature algorithm such as the Digital Signature Algorithm (DSA) satisfies the last three. In this thesis, we focus on the second goal – authenticity.

Secure authentication is a need in many different areas. Smartcards, Internet of things devices, and product packaging are a few examples. The goal, in general, is to prove the authenticity of information, e.g. proving the ownership of a bank account. The conventional way of doing so is to store a cryptographic key in non-volatile memory and use it in combination with digital signature algorithms such as (Elliptic Curve) DSA or the RSA signature. Thus, in the area of mobile authentication devices, attacks are usually not directed to the cryptographic primitives used by the system but aim for the extraction of the secret key stored in it. The methods range from power analysis to invasive attacks where the attacker tries to physically extract the key from a non-volatile memory [Sko05]. Attacks on popular commercial systems become known over and over again [Eis+08; KSP10]. As attackers develop new techniques to extract keys, designers and attackers find themselves in an everlasting race for supremacy. Thus, the costs of securing a key are increasing since several countermeasures have to be applied to prevent the improved techniques used by the attackers from succeeding. To counteract this steady increase of costs a new approach to secure authentication emerged as a potential lightweight alternative to the conventional methods.

The concept of *Physical Unclonable Functions* (PUFs) was first introduced by Pappu et. al. in 2002 [Pap+02] under the name of Physical One-Way Functions. They pointed a laser beam onto a physical semi-transparent 3D structure creating unique speckle patterns for different angles. The patterns are then transformed to a bit string representing the PUF response. Because even the slightest alteration of the structure would lead to a different behavior of the PUF it is unlikely for an adversary to forge the responses, since, in theory, he would have to flawlessly clone the physical structure on a nanoscale level.

PUFs, therefore, stand in opposition to classic tamper-resistant authentication devices where the key is stored on a chip and several countermeasures are applied to prevent extraction followed by duplication of the key. Since a PUF does not contain a key the authentication is done with *Challenge-Response-Pairs* (CRPs), i.e. a bit string is sent to the PUF which in turn responds with one or multiple bits. During the initialization phase, the entity that wants to authenticate the PUF collects and stores an amount of CRPs. Afterward, it is able to verify the validity of the PUF by querying the responses of multiple known challenges and comparing them with the stored ones. In Pappu's PUF, for instance, the challenge would be the angle of the laser and the response the transformed laser speckle patterns resulting from this specific angle.

The concept of PUFs aims for a lightweight and low-cost authentication device that only can be used by the person in physical possession of it. The idea is to use the unique physical properties of a carrier medium to create different behaving functions using the same blueprint. In Pappu's PUF the resulting speckle patterns were used as the unique property. All PUF instances of this concept are built the same way but due to varieties in the different semi-transparent 3D structures, they all behave distinctively. This type of PUF is categorized as a so-called optical PUF. The drawbacks of this PUF are that it requires a high precision setup for a reliable generation of the laser speckles. This renders them impractical for casual usage, as this high precision cannot be guaranteed in an everyday life situation. Consequently, the research has shifted in the direction of the design of so-called electrical PUFs. They use the special properties of integrated circuits, e.g. signal delay variations, to derive a secure authentication procedure. In contrast to Pappu's optical PUF they do not need laboratory conditions to reliably work and therefore potentially represent a real world solution. An example for electrical PUFs is the SRAM-PUF [Gua+07] which uses RAM cells and performs an incomplete store operation. Each cell requires a distinct duration to complete this operation such that finally, a bitstring results from the actual values stored in the cells. Another example is the Ring Oscillator PUF [SD07] which uses simple circuits that oscillate at a certain frequency, which is caused by manufacturing variations. The output bits are then generated by comparing the frequencies of two oscillators determined by the challenge. As the final example, there is the Arbiter PUF [Gas+02] on which we focus in this thesis. It generates output bits by evaluating a signal race condition where the signal paths are fixed by the challenge.

We use the terms of Strong PUF and Weak PUF to distinguish between two different types of PUFs [Gua+07]. Weak PUFs have a small amount (i.e. polynomial in input length) of potential CRPs and therefore they need additional restrictive measures such that an attacker is not able to read out all CRPs, which would effectively result in a clone of the PUF. Strong PUFs, however, must have an exponentially large challenge space and hence an attacker is not efficiently able to clone the PUF by exhausting all potential CRPs. They have the property that even the manufacturer is not able to create a counterfeit of the PUF making it possible to build the PUF in an untrusted environment. In general, if an adversary gets hold of a PUF instance he must not be able to clone the device's behavior efficiently (i.e. in polynomial time). A Strong PUF satisfying these properties, however, is not known yet. Taking, for instance, the Arbiter PUF it was noted from the initial authors that due to its linear delay model it may be susceptible to Machine Learning (ML) attacks. Shortly after the Arbiter PUF was proposed Lim used this fact to show that it can indeed be efficiently cloned using linear Support Vector Machines [Lim04]. Nevertheless, the Arbiter PUF has become popular because of its lightweight design and potentially large challenge space size. Several attempts were made to enhance its security, e.g. by XOR combining multiple instances to add non-linearity, and also transforming the inputs of individual PUFs. Lately, apart from the Arbiter PUF, efforts were put into finding new PUF candidates exploiting non-linear properties of electrical circuits. From this, for instance, a PUF based on properties of current mirrors emerged [KB14] which, though, has been broken shortly after [Guo+16]. Another approach is to use non-linear voltage transfer properties for a new type of PUF [VK15], which by now has not been attacked successfully.

When we look Arbiter PUFs we observe that, not considering the effects of environmental noise, their underlying behavior can be described as a Boolean function. The PUF obtains multiple bits as input and outputs a single bit. Boolean functions are a well-known field of research. Especially in the domain of cryptography, the analysis of Boolean functions is an important technique, for instance concerning their algebraic immunity [DGM05]. An important tool for the analysis of Boolean functions is their Fourier expansion which represents a function as a multilinear polynomial with real coefficients. It was first introduced by Walsh [Wal23]. The goal of this thesis is to use this technique to find characteristics of the Arbiter PUF. We will, therefore, utilize the notion of influence which was first used by Penrose [Pen46]. As we will see later this notion is directly related to the Fourier expansion and we will use it to make statements about the learnability of Arbiter PUFs.

In the next chapter, we start by giving a detailed introduction of the Fourier expansion and Arbiter PUF. We will formalize various aspects of the Arbiter PUF as well as the XOR Arbiter PUF and present the current state-of-the-art attacks. After that, we will use these formalizations as the basis for the influence analysis and finally give a conclusion.

2 PRELIMINARIES

In this chapter we give an overview of the Fourier expansion and Arbiter PUF. We show how the Arbiter PUF as well as the XOR Arbiter PUF can be described mathematically and introduce the notion of stability. Finally we present the current state-of-the-art attacks on these types of PUFs.

2.1 FOURIER EXPANSION

This thesis revolves around Boolean functions following the notation from O'Donnell [ODo14],

$$f : \{-1, 1\}^n \to \{-1, 1\}.$$

In this case we used $\{-1, 1\}$ to represent our binary values (bits). This can be replaced by $\mathbb{F}_2$ or $\{0, 1\}$ as required. It is even possible and often useful to look at the more general real valued Boolean functions $f : \{-1, 1\}^n \to \mathbb{R}$.

Every Boolean function can be uniquely expressed as a multilinear polynomial over $\mathbb{R}$, i.e. the highest potency in a monomial is 1. Bigger exponents for $x \in \{-1, 1\}$ are not possible anyway since $x^k = 1$ for even k and $x^k = x$ for odd k. This polynomial representation is called the *Fourier expansion* of f. We define the coefficients as well as the monomials of the polynomial on subsets S of the index-set $[n] = \{1, 2, \dots, n\}$. The coefficients (which we call *Fourier coefficients*) are denoted as $\widehat{f}(S) \in \mathbb{R}$. The monomials are represented by the parity functions $\chi_S : \{-1, 1\}^n \to \{-1, 1\}$ with

$$\chi_S(x) = \prod_{i \in S} x_i. \qquad \text{(where } \chi_\varnothing(x) = 1\text{)}$$

We hence write the Fourier expansion of the function f as

$$f(x) = \sum_{S \subseteq [n]} \widehat{f}(S)\chi_S(x).$$

Looking, for instance, at the Fourier expansion of a parity function χ_S we can easily notice that the Fourier coefficients $\widehat{f}(T)$ are zero except for $T = S$ which is 1. Next we define an inner product on functions.

Definition 2.1. The inner product $\langle \cdot, \cdot \rangle$ of two functions $f, g : \{-1, 1\}^n \to \mathbb{R}$ is defined as

$$\langle f, g \rangle = 2^{-n} \sum_{x \in \{-1,1\}^n} f(x)g(x) = \mathop{\mathbf{E}}_{\boldsymbol{x} \sim \{-1,1\}^n} [f(\boldsymbol{x})g(\boldsymbol{x})].$$

The linearity of the inner product follows from this definition. We write $\boldsymbol{x} \sim \{-1,1\}^n$ to specify that $\boldsymbol{x}$ is a random variable where each $\boldsymbol{x}_i$ is drawn independently and uniformly at random from $\{-1,1\}$. Analogously we write $\boldsymbol{x} \sim \mathcal{N}\left(\mu,\sigma^2\right)$ to denote that $\boldsymbol{x}$ is drawn from a Gaussian distribution with mean μ and variance σ^2. Random variables are always displayed in **bold-font**. Equipped with this definition we can state our first Theorem.

Theorem 2.2. *The inner product of two parity functions χ_S and χ_T is*

$$\langle \chi_S, \chi_T \rangle = \begin{cases} 1 & \text{if } S = T, \\ 0 & \text{if } S \neq T. \end{cases}$$

Proof. The first step is to show that $\chi_S(x)\chi_T(x) = \chi_{S\Delta T}(x)$ where $S\Delta T = \{i \mid i \in S \cup T \wedge i \notin S \cap T\}$ is the symmetric difference.

$$\chi_S(x)\chi_T(x) = \prod_{i\in S} x_i \prod_{i \in T} x_i = \prod_{i\in S\Delta T} x_i \prod_{i \in S\cap T} x_i^2 = \prod_{i\in S\Delta T} x_i = \chi_{S\Delta T}(x)$$

The essential step here is that all elements in the symmetric difference only appear once in the product. The elements from the intersection however appear twice. The second step is to show that

$$\mathop{\mathbf{E}}_{\boldsymbol{x}}\left[\chi_S(\boldsymbol{x})\right] = \begin{cases} 1 & \text{if } S = \varnothing, \\ 0 & \text{if } S \neq \varnothing. \end{cases}$$

For $S = \varnothing$ we know $\mathbf{E}_{\boldsymbol{x}}\left[\chi_S(\boldsymbol{x})\right] = \mathbf{E}_{\boldsymbol{x}}\left[1\right] = 1$. For the other case we have

$$\mathop{\mathbf{E}}_{\boldsymbol{x}}\left[\chi_S(\boldsymbol{x})\right] = \mathop{\mathbf{E}}_{\boldsymbol{x}}\left[\prod_{i\in S}\boldsymbol{x}_i\right] = \prod_{i\in S}\mathop{\mathbf{E}}_{\boldsymbol{x}}\left[\boldsymbol{x}_i\right] = 0,$$

since we choose all $\boldsymbol{x}_i$ independently from each other. As they are drawn uniformly at random from $\{-1,1\}$ their expected value is zero. Combining everything we have

$$\langle \chi_S, \chi_T \rangle = \mathop{\mathbf{E}}_{\boldsymbol{x}}\left[\chi_S(\boldsymbol{x})\chi_T(\boldsymbol{x})\right] = \mathop{\mathbf{E}}_{\boldsymbol{x}}\left[\chi_{S\Delta T}(\boldsymbol{x})\right] = \begin{cases} 1 & \text{if } S = T, \\ 0 & \text{if } S \neq T, \end{cases}$$

since $S\Delta T = \varnothing \Leftrightarrow S = T$. □

If we identify a Boolean function by its outputs placed in a 2^n dimensional vector, we can also write it as a linear combination of its Fourier coefficients and the parity function vectors. Hence, from this linear algebraic point of view the theorem shows that the functions χ_S are all orthogonal to each other. They indeed form an orthonormal basis of the vector space of real valued Boolean functions. This also justifies the Fourier expansion's existence as well as uniqueness, since the parity functions are pairwise linearly independent. With this knowledge we can state a formula for the computation of the Fourier coefficients.

Theorem 2.3. *For a Boolean function $f : \{-1,1\}^n \to \mathbb{R}$ and a subset $S \subseteq [n]$ the Fourier coefficient can be determined by*

$$\widehat{f}(S) = \langle f, \chi_S \rangle = \mathop{\mathbf{E}}_{\boldsymbol{x} \sim \{-1,1\}^n} \left[f(\boldsymbol{x}) \chi_S(\boldsymbol{x}) \right].$$

Proof.

$$\langle f, \chi_S \rangle = \left\langle \sum_{T \subseteq [n]} \widehat{f}(T) \chi_T, \chi_S \right\rangle = \sum_{T \subseteq [n]} \widehat{f}(T) \langle \chi_T, \chi_S \rangle = \widehat{f}(S)$$

We use the definition of the Fourier expansion, the linearity of the inner product and Theorem 2.2. □

This however will be hard to do in practice since the amount of terms in the sum grows exponentially in the input length. We can approximate Fourier coefficients by calculating the average of a smaller set of terms. To that end, we need a special case of Hoeffding's inequality [Hoe63].

Lemma 2.4. *(Hoeffding) Let $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ be random variables drawn independently from the same distribution with $\boldsymbol{x}_i \in [-1,1]$ and $\boldsymbol{S} = \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{x}_i$. Then for $\lambda > 0$,*

$$\mathop{\mathbf{Pr}}_{\boldsymbol{x}_i} \left[\left| \boldsymbol{S} - \mathop{\mathbf{E}}_{\boldsymbol{x}_i} [\boldsymbol{S}] \right| \geq \lambda \right] \leq 2 e^{-\lambda^2 n / 2}.$$

Proof. From Theorem 2 by Hoeffding [Hoe63] for $a_i \leq \boldsymbol{x}_i \leq b_i$ we have

$$\mathop{\mathbf{Pr}}_{\boldsymbol{x}_i} \left[\boldsymbol{S} - \mathop{\mathbf{E}}_{\boldsymbol{x}_i} [\boldsymbol{S}] \geq \lambda \right] \leq e^{-2\lambda^2 n^2 / \sum_{i=1}^{n} (b_i - a_i)^2}.$$

Since in our case $-1 \leq \boldsymbol{x}_i \leq 1$ we get

$$\begin{aligned} \mathop{\mathbf{Pr}}_{\boldsymbol{x}_i} \left[\boldsymbol{S} - \mathop{\mathbf{E}}_{\boldsymbol{x}_i} [\boldsymbol{S}] \geq \lambda \right] &\leq e^{-2\lambda^2 n^2 / 4n} \\ &\leq e^{-\lambda^2 n / 2}. \end{aligned}$$

Finally, we add the probabilities for both ends, i.e.

$$\mathop{\mathbf{Pr}}_{\boldsymbol{x}_i} \left[\left| \boldsymbol{S} - \mathop{\mathbf{E}}_{\boldsymbol{x}_i} [\boldsymbol{S}] \right| \geq \lambda \right] = \mathop{\mathbf{Pr}}_{\boldsymbol{x}_i} \left[\boldsymbol{S} - \mathop{\mathbf{E}}_{\boldsymbol{x}_i} [\boldsymbol{S}] \geq \lambda \right] + \mathop{\mathbf{Pr}}_{\boldsymbol{x}_i} \left[\mathop{\mathbf{E}}_{\boldsymbol{x}_i} [\boldsymbol{S}] - \boldsymbol{S} \geq \lambda \right].$$

We define $\boldsymbol{x}_i' = -\boldsymbol{x}_i$ which still satisfy $\boldsymbol{x}_i' \in [-1,1]$. Defining $\boldsymbol{S}' = \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{x}_i'$ we can hence write

$$\Pr_{\boldsymbol{x}_i}\left[\left|\boldsymbol{S} - \mathop{\mathbf{E}}_{\boldsymbol{x}_i}[\boldsymbol{S}]\right| \geq \lambda\right] = \Pr_{\boldsymbol{x}_i}\left[\boldsymbol{S} - \mathop{\mathbf{E}}_{\boldsymbol{x}_i}[\boldsymbol{S}] \geq \lambda\right] + \Pr_{\boldsymbol{x}_i}\left[\boldsymbol{S}' - \mathop{\mathbf{E}}_{\boldsymbol{x}_i}[\boldsymbol{S}'] \geq \lambda\right]$$
$$\leq 2e^{-\lambda^2 n/2}.$$

□

Next we formulate an important fact about the Fourier spectrum of a function f using the inner product which is called Parseval's Theorem.

Theorem 2.5. *(Parseval's Theorem) For a function $f : \{-1,1\}^n \to \mathbb{R}$,*

$$\langle f, f\rangle = \mathop{\mathbf{E}}_{\boldsymbol{x}}\left[f(\boldsymbol{x})^2\right] = \sum_{S\subseteq[n]} f(S)^2.$$

Hence, for a Boolean valued function $f : \{-1,1\}^n \to \{-1,1\}$,

$$\sum_{S\subseteq[n]} \widehat{f}(S)^2 = 1,$$

since $\forall_x : f(x)^2 = 1$.

Proof. We can verify this in the same manner we did before:

$$\langle f, f\rangle = \left\langle \sum_{S\subseteq[n]} \widehat{f}(S)\chi_S, \sum_{T\subseteq[n]} \widehat{f}(T)\chi_T \right\rangle$$
$$= \sum_{S,T\subseteq[n]} \widehat{f}(S)\widehat{f}(T)\langle \chi_S, \chi_T\rangle = \sum_{S\subseteq[n]} \widehat{f}(S)^2$$

Again we use Theorem 2.2, which eliminates all addends with $S \neq T$. □

We can also analogously state this theorem for any two functions $f, g : \{-1,1\}^n \to \mathbb{R}$. This generalization is called Plancherel's Theorem.

Theorem 2.6. *(Plancherel's Theorem)*
For any two functions $f, g : \{-1,1\}^n \to \mathbb{R}$,

$$\langle f, g\rangle = \mathop{\mathbf{E}}_{\boldsymbol{x}}\left[f(\boldsymbol{x})g(\boldsymbol{x})\right] = \sum_{S\subseteq[n]} \widehat{f}(S)\widehat{g}(S).$$

The proof works equivalently to Theorem 2.5, replacing the second f by g. The inner product can also be used to measure the similarity between two functions f and g which we call the distance.

Definition 2.7. The distance between two functions $f, g : \{-1,1\}^n \to \{-1,1\}$ is defined as

$$\operatorname{dist}(f, g) = \Pr_{\boldsymbol{x}}\left[f(\boldsymbol{x}) \neq g(\boldsymbol{x})\right].$$

If we now look at the inner product of two Boolean valued functions f, g we can see that $f(x)g(x)$ is 1 if $f(x) = g(x)$ and -1 if $f(x) \neq g(x)$. Using the definition of the expected value we can write the following.

Proposition 2.8. *For $f, g : \{-1,1\}^n \to \{-1,1\}$,*

$$\langle f, g\rangle = \Pr_{\boldsymbol{x}}[f(\boldsymbol{x}) = g(\boldsymbol{x})] - \Pr_{\boldsymbol{x}}[f(\boldsymbol{x}) \neq g(\boldsymbol{x})] = 1 - 2\text{dist}(f, g).$$

The next important part especially for the connection to Arbiter PUFs are threshold functions.

Definition 2.9. A function $f : \{-1,1\}^n \to \{-1,1\}$ is called a linear threshold function (LTF) if it can be expressed as

$$f(x) = \text{sgn}\left(a_0 + a_1x_1 + a_2x_2 + ... + a_nx_n\right),$$

with $a_0, a_1, ..., a_n \in \mathbb{R}$.

As we can see, an LTF is characterized by $n+1$ variables and can also be described as a degree-1 polynomial packaged into the sgn operator. It can also be seen as a weighted majority. We define $\text{sgn}(0) = 1$, but in fact every LTF can have its weight adjusted by a small portion such that the outputs do not change and the case $\text{sgn}(0)$ does not appear. We extend this notion to higher degree polynomials.

Definition 2.10. A function $f : \{-1,1\}^n \to \{-1,1\}$ is called a polynomial threshold function (PTF) of degree at most k if it can be expressed as $f(x) = \text{sgn}\left(p(x)\right)$ where $p : \{-1,1\}^n \to \mathbb{R}$ is a polynomial of degree at most k with real coefficients.

2.2 ARBITER PUF

The Arbiter PUF was first introduced by Gassend et. al. in 2002 [Gas+02]. It consists of n stages where n is the amount of input bits. Each stage has an upper and a lower path. To start the evaluation for a given challenge, two signals are introduced to the first stage of the PUF. The signals subsequently pass through each stage whereby the challenge bit decides whether the signal paths to the next stage are crossed or left straight. In this process each stage delays the signals by a distinct amount. The delay values are caused by manufacturing imperfections which make each PUF unique. Finally the signals arrive at the last component, the arbiter, which determines on which path the signal arrives first and sets the output bit accordingly. Note that it does not matter on which path the fastest signal started in the first place. It only matters on which input path of the arbiter it arrives. An illustration can be seen in Figure 2.2.1.

2.2.1 *Model*

To understand how the Arbiter PUF works we want to find a mathematical model to describe it. Our goal is, given an Arbiter PUF instance $p : \{-1,1\}^n \to \{-1,1\}$, to find a model $f : \{-1,1\}^n \to \{-1,1\}$

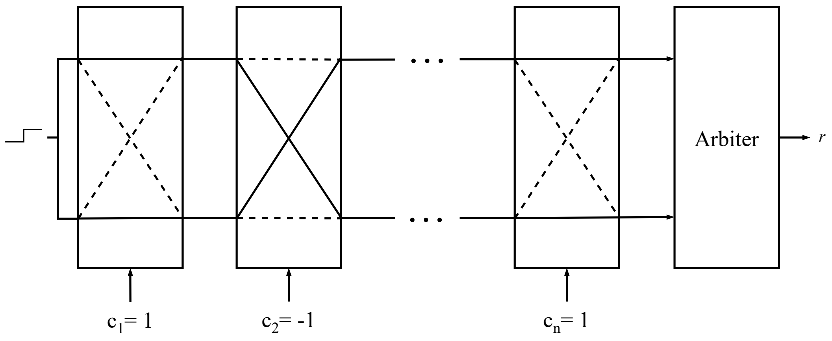

Figure 2.2.1: Scheme of an Arbiter PUF

of the PUF which satisfies $\text{dist}(p, f) = 0$. Arbiter PUFs generally are no mathematical functions since the outputs can be different for the same challenge due to noise. However, we want our model to describe the PUFs behavior when not considering noise.

Our first step is to define the values $\delta_{c_i}^{(i)}$ for each stage $1 \leq i \leq n$ we where c_i is the input bit at position i. These values represent the delay differences introduced by the stage for the crossed and straight propagation respectively. We define delay differences instead of absolute path delay values since in the end, the arbiter decides the output bit depending on the difference anyway. The value of the delay difference is positive if the upper path was faster and negative in the opposite case. Additionally, using the delay differences instead of the absolute path runtime values reduces the required number of variables to describe a PUF instance. If we add up all the delay differences, the sign of the result shows which path arrived first and therefore equals the output, i.e. 1 for the upper and -1 for the lower path. However, if a delay is introduced in some stage i and later at stage $j > i$ the paths are crossed, then the sign of the delay has to be changed since the impact of the delay difference is inverted. Thus for a stage i, it has to be done for every stage j, with $i \leq j \leq n$ if the paths are crossed. Hence we can model the Arbiter PUF as follows:

$$f(c) = \text{sgn}\left(\sum_{i=1}^{n}\left(\delta_{c_i}^{(i)}\prod_{j=i}^{n} c_j\right)\right). \tag{2.2.1}$$

This formula can be simplified in a matter that the delay difference variables are no longer dependent on the input. For this, we show the following statement.

Lemma 2.11. *The delay difference $\delta_{c_i}^{(i)}$ can be expressed as a function of c_i, i.e.*

$$\delta_{c_i}^{(i)} = \frac{1}{2}\left[\left(\delta_1^{(i)} - \delta_{-1}^{(i)}\right)c_i + \delta_1^{(i)} + \delta_{-1}^{(i)}\right].$$

This can easily be proven by evaluating the function for $c_i = 1$ and $c_i = -1$ respectively. Equipped with this fact we can show the correctness of the simplification.

Theorem 2.12. *An Arbiter PUF $f : \{-1,1\}^n \to \{-1,1\}$ with*

$$f(c) = \operatorname{sgn}\left(\sum_{i=1}^{n}\left(\delta_{c_i}^{(i)}\prod_{j=i}^{n} c_j\right)\right)$$

can be expressed as a PTF $g : \{-1,1\}^n \to \{-1,1\}$ with

$$g(c) = \operatorname{sgn}\left(\sum_{i=1}^{n}\left(w_i\prod_{j=i}^{n} c_j\right) + w_0\right),$$

where

$$w_i = \begin{cases} \frac{1}{2}\left(\delta_1^{(n)} - \delta_{-1}^{(n)}\right) & \text{if } i = 0, \\ \frac{1}{2}\left(\delta_1^{(i)} + \delta_{-1}^{(i)}\right) & \text{if } i = 1, \\ \frac{1}{2}\left(\delta_1^{(i)} + \delta_{-1}^{(i)} + \delta_1^{(i-1)} - \delta_{-1}^{(i-1)}\right) & \text{if } i > 1. \end{cases}$$

Proof. The proof is done by induction on n with $f(c) = \operatorname{sgn}(p(c))$and $g(c) = \operatorname{sgn}(q(c))$. It will be shown that $p(c) = q(c)$ and therefore $f(c) = g(c)$. Since w_0 and w_1 are special cases, we prove the basis for $n = 1$ and $n = 2$.
Basis: The statement holds for $n = 1$.

$$\begin{aligned} p(c) = \delta_{c_1}^{(1)} c_1 &= \frac{1}{2}\left[\left(\delta_1^{(1)} - \delta_{-1}^{(1)}\right)c_1 + \delta_1^{(1)} + \delta_{-1}^{(1)}\right]c_1 \\ &= \frac{1}{2}\left(\delta_1^{(1)} - \delta_{-1}^{(1)}\right) + \frac{1}{2}\left(\delta_1^{(1)} + \delta_{-1}^{(1)}\right)c_1 = w_0^{(1)} + w_1 c_1 = q(c), \end{aligned}$$

where the second equality uses Lemma 2.11 and $w_0^{(i)} = \frac{1}{2}\left(\delta_1^{(i)} - \delta_{-1}^{(i)}\right)$.
The statement holds for $n = 2$.

$$\begin{aligned} p(c) = \delta_{c_1}^{(1)} c_1 c_2 + \delta_{c_2}^{(2)} c_2 &= \frac{1}{2}\left[\left(\delta_1^{(2)} - \delta_{-1}^{(2)}\right)c_2 + \delta_1^{(2)} + \delta_{-1}^{(2)}\right]c_2 \\ &\qquad \text{(Lemma 2.11)} \\ &= \delta_{c_1}^{(1)} c_1 c_2 + \frac{1}{2}\left(\delta_1^{(2)} - \delta_{-1}^{(2)}\right) + \frac{1}{2}\left(\delta_1^{(2)} + \delta_{-1}^{(2)}\right)c_2 \\ &= \frac{1}{2}\left[\left(\delta_1^{(1)} - \delta_{-1}^{(1)}\right)c_1 + \delta_1^{(1)} + \delta_{-1}^{(1)}\right]c_1 c_2 + w_0^{(2)} + \frac{1}{2}\left(\delta_1^{(2)} + \delta_{-1}^{(2)}\right)c_2 \\ &\qquad \text{(Lemma 2.11)} \\ &= \frac{1}{2}\left(\delta_1^{(1)} - \delta_{-1}^{(1)}\right)c_2 + \frac{1}{2}\left(\delta_1^{(1)} + \delta_{-1}^{(1)}\right)c_1 c_2 + w_0^{(2)} + \frac{1}{2}\left(\delta_1^{(2)} + \delta_{-1}^{(2)}\right)c_2 \\ &= w_2 c_2 + w_1 c_1 c_2 + w_0^{(2)} = q(c) \end{aligned}$$

Inductive Step: If the statement holds for n it also holds for $n+1$.

$$\begin{aligned}
q(c) &= \sum_{i=1}^{n+1}\left(w_i\prod_{j=i}^{n+1}c_j\right) + w_0^{(n+1)} \\
&= c_{n+1}\left(\sum_{i=1}^{n}\left(w_i\prod_{j=i}^{n}c_j\right) + w_{n+1}\right) + w_0^{(n+1)} \\
&= c_{n+1}\left(\sum_{i=1}^{n}\left(w_i\prod_{j=i}^{n}c_j\right) + \frac{1}{2}\left(\delta_{-1}^{(n+1)} + \delta_{1}^{(n+1)}\right) + \frac{1}{2}\left(\delta_{1}^{(n)} - \delta_{-1}^{(n)}\right)\right) + w_0^{(n+1)} \\
&= c_{n+1}\left(\sum_{i=1}^{n}\left(w_i\prod_{j=i}^{n}c_j\right) + w_0^{(n)} + \frac{1}{2}\left(\delta_{-1}^{(n+1)} + \delta_{1}^{(n+1)}\right)\right) + w_0^{(n+1)} \\
&\overset{I.H.}{=} c_{n+1}\left(\sum_{i=1}^{n}\left(\delta_{c_i}^{(i)}\prod_{j=i}^{n}c_j\right) + \frac{1}{2}\left(\delta_{-1}^{(n+1)} + \delta_{1}^{(n+1)}\right)\right) + w_0^{(n+1)} \\
&= \sum_{i=1}^{n}\left(\delta_{c_i}^{(i)}\prod_{j=i}^{n+1}c_j\right) + \frac{1}{2}\left(\delta_{-1}^{(n+1)} + \delta_{1}^{(n+1)}\right)c_{n+1} + \frac{1}{2}\left(\delta_{1}^{(n+1)} - \delta_{-1}^{(n+1)}\right) \\
&= \sum_{i=1}^{n}\left(\delta_{c_i}^{(i)}\prod_{j=i}^{n+1}c_j\right) + \delta_{c_{n+1}}^{(n+1)}c_{n+1} = p(c),
\end{aligned}$$

where a variant of Lemma 2.11 is once more used in the penultimate equality. □

Since we later state theorems using different representations of the Arbiter PUF we explicitly name them.

Definition 2.13. We call a Boolean function $f : \{-1,1\}^n \to \{-1,1\}$ with

$$f(c) = \operatorname{sgn}\left(\sum_{i=1}^{n}\left(w_i\prod_{j=i}^{n}c_j\right) + w_0\right)$$

for $w_0,\dots,w_n \in \mathbb{R}$ an *Arbiter Threshold Function* (ATF).

As we can see the ATF only has $n+1$ monomials, hence we can transform it into an LTF.

Theorem 2.14. *An ATF $f : \{-1,1\}^n \to \{-1,1\}$ with*

$$f(c) = \operatorname{sgn}\left(\sum_{i=1}^{n}\left(w_i\prod_{j=i}^{n}c_j\right) + w_0\right)$$

can be expressed as an LTF $g : \{-1,1\}^n \to \{-1,1\}$ with

$$g(x) = \operatorname{sgn}\left(\sum_{i=i}^{n}w_ix_i + w_0\right),$$

where $t(c) = x$ with $x_i = \prod_{j=i}^{n}c_j$ is a bijection.

Proof. We need to show that the transformation $t : \{-1,1\}^n \to \{-1,1\}^n$ with $t(c) = x$ defined by $x_i = \prod_{j=i}^n c_j$ is a bijection.

We claim the inverse transformation is $t^{-1} : \{-1,1\}^n \to \{-1,1\}^n$ with $t^{-1}(x) = c$ defined by $c_i = x_i \cdot x_{i+1}$, setting $x_{n+1} = 1$. To check the validity of the inverse transformation we compute $t(t^{-1}(x))$, i.e.

$$t(t^{-1}(x))_i = \prod_{j=i}^{n} x_j \cdot x_{j+1} = x_i.$$

Since a valid inverse exists the transformation t is a bijection. □

2.2.2 *Stability & Noise*

By now we modeled the Arbiter PUF as a mathematical function, that is we get a determined output for a fixed input. Although this is the desired behavior, in reality an Arbiter PUF is influenced by environmental conditions such as fluctuations of temperature or voltage. This leads to random alterations of the delay differences of the PUF. We define the notion of a noise affected PUF as follows.

Definition 2.15. Let $p : \{-1,1\}^n \to \{-1,1\}$ be the model of a PUF. We call $\mathring{p} : \{-1,1\}^n \times \mathbb{R}^k \to \{-1,1\}$ the noisy model of this PUF, satisfying $\mathring{p}\,(x,0) = p\,(x)$ for all $x \in \{-1,1\}^n$.

Following this, we can define the notion of stability which quantifies the impact of the noise on a PUF.

Definition 2.16. Let $p : \{-1,1\}^n \to \{-1,1\}$ be the model of a PUF. For a challenge $c \in \{-1,1\}^n$ we define the stability as

$$\text{Stab}_p\,(c) = \mathop{\mathbf{Pr}}_{\varepsilon_i \sim D_i} \left[p(c) = \mathring{p}(c,\varepsilon)\right],$$

where the D_i are the PUF specific noise distributions and the ε_i are drawn independently.

Looking at Arbiter PUFs as analyzed by Delvaux and Verbauwhede [DV13] we can divide the delay difference into two parts, i.e. the model delay difference $\Delta_{\text{Model}}(c)$ as stated inside the sgn operator in Theorem 2.12 and a noise part represented by a single real-valued parameter Δ_{Noise}. The noisy Arbiter PUF model $\mathring{f} : \{-1,1\}^n \times \mathbb{R} \to \{-1,1\}$ can hence be expressed as

$$\mathring{f}\,(c, \Delta_{\text{Noise}}) = \text{sgn}\,(\Delta_{\text{Model}}(c) + \Delta_{\text{Noise}})\,, \qquad (2.2.2)$$
$$\text{where } \Delta_{\text{Model}}(c) = \sum_{i=1}^{n} \left(w_i \prod_{j=i}^{n} c_j \right) + w_0.$$

The Arbiter PUF's specific noise distribution is a normal distribution, i.e. $\Delta_{\text{Noise}} \sim \mathcal{N}\left(0, \sigma^2_{\text{Noise}}\right)$. Since an additional parameter is introduced, the PUF now may not return the same response for the same challenge anymore. Applying the term of stability to the Arbiter PUF we obtain the following.

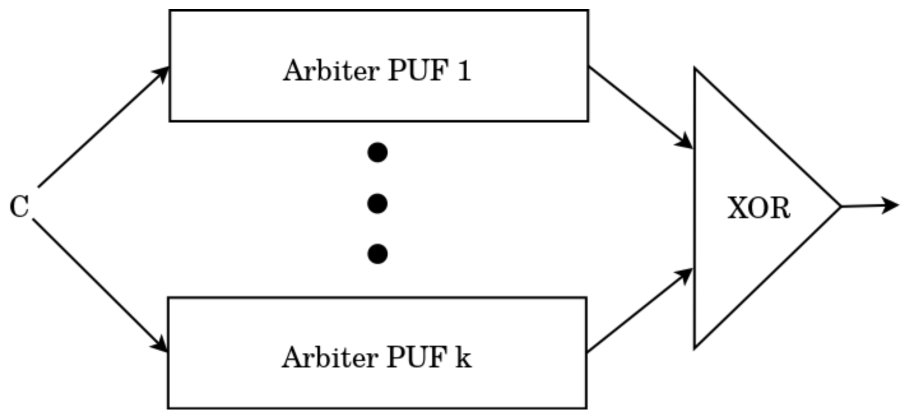

Figure 2.3.1: Illustration of the XOR Arbiter PUF

Corollary 2.17. *Let $\mathring{f} : \{-1,1\}^n \times \mathbb{R} \to \{-1,1\}$ be the noisy Arbiter PUF model with $\mathring{f}(c, \Delta_{\text{Noise}}) = \operatorname{sgn}(\Delta_{\text{Model}}(c) + \Delta_{\text{Noise}})$ as above. Then for a fixed challenge $c \in \{-1,1\}^n$,*

$$\operatorname{Stab}_{\mathring{f}}(c) = \Pr_{\Delta_{\text{Noise}}}\left[\operatorname{sgn}(\Delta_{\text{Model}}(c)) = \operatorname{sgn}(\Delta_{\text{Model}}(c) + \Delta_{\text{Noise}})\right].$$

It becomes evident that, for instance, the probability for the output of the PUF to flip is higher for challenges c which have $\Delta_{Model}(c)$ with small magnitudes compared to Δ_{Noise}.

2.3 XOR ARBITER PUF

As it became known that due to its simple linear delay model an Arbiter PUF is susceptible to ML attacks [Lim04], efforts were put into the direction of extending the Arbiter PUF to a more secure construct.

To counteract the simplicity of the Arbiter PUF's linear characterization the idea of XORing the outputs of a single Arbiter PUF was first proposed in 2007 [SD07]. Devadas extended this approach to the combination of multiple Arbiter PUFs [Dev09]. The goal was to add nonlinearity[1] to the PUF to prevent simple linear separation based ML attacks. The XOR Arbiter PUF takes a challenge $c \in \{-1,1\}^n$ and passes it to all k Arbiter PUFs. The outputs of the Arbiter PUFs are then XORed and form the final response of the PUF (see Figure 2.3.1).

1 Note that the notion of linearity depends on the vector spaces in which the function operates. The XOR operation $f : \mathbb{F}_2^n \to \mathbb{F}_2$ with $f(x) = \bigoplus_{i=1}^n x_i$ is a linear function. Using $\Phi : \mathbb{F}_2^n \to \mathbb{R}$ with $\Phi(x)_i = 1$ if $x_i = 0$ and $\Phi(x)_i = -1$ if $x_i = 1$, we naturally extend the XOR operation to $g : \mathbb{R}^n \to \mathbb{R}$ with $g(x) = \prod_{i=1}^n x_i$, which is non-linear.

2.3.1 *Model*

The XOR operation (parity function) on $\{-1,1\}$ can be represented by classical multiplication. If we take the product of k ATFs then, using the fact that for $a_1,\ldots,a_l \in \mathbb{R}$,

$$\operatorname{sgn}\left(\prod_{i=1}^{l} a_i\right) = \prod_{i=1}^{l} \operatorname{sgn}(a_i),$$

we can collapse the sgn operators, giving us

$$f_{XOR}(x) = \operatorname{sgn}\left(\prod_{l=1}^{k}\left(\sum_{i=1}^{n}\left(w_i^{(l)}\prod_{j=i}^{n} c_j\right)\right)\right);$$

applying the transformation as in Theorem 2.14 we get

$$f_{XOR}(x) = \operatorname{sgn}\left(\prod_{l=1}^{k}\left(\sum_{i=1}^{n}\left(w_i^{(l)} x_i\right)\right)\right). \tag{2.3.1}$$

As we can see, the XOR Arbiter PUF cannot be modeled as an LTF with dimension n since the product of the individual Arbiter PUFs produces more than $n+1$ different monomials. However, since each individual PUF can still be represented by its weight vector the number of variables on which the PUF depends grows only linearly in k. Furthermore, it is possible to linearize the PTF resulting from the expansion of the product, giving an LTF representation with dimension $\mathcal{O}\left(n^k\right)$.

2.3.2 *Stability*

A problem of the XOR Arbiter PUF is that for large k the construction of the design is infeasible. This is due to the amplification of the single Arbiter PUF's noise by the XOR operation. In fact, the noise of the XOR Arbiter PUF grows exponentially in k. We will need the following lemma to prove this statement.

Lemma 2.18. *For $a,b,c \in \mathbb{R}$, let $f : \mathbb{N}_0 \to \mathbb{R}$ be a first order non-homogeneous recurrence relation defined by*

$$f(n) = \begin{cases} af(n-1)+b & \text{if } n>0,\\ c & \text{if } n=0.\end{cases}$$

Then for $a \neq 1$ the relation can be expressed as

$$f(n) = a^n c + \frac{a^n-1}{a-1}b.$$

Proof. We will prove this by induction on n.

Basis: The statement holds for $n = 0$,

$$\begin{aligned} f(0) &= a^0 c + \frac{a^0 - 1}{a-1} b \\ &= c \end{aligned}$$

and the statement holds for $n = 1$,

$$\begin{aligned} f(1) &= a^1 c + \frac{a^1 - 1}{a-1} b \\ &= ac + b. \end{aligned}$$

Inductive Step: If the statement holds for n it also holds for $n+1$.

$$\begin{aligned} f(n+1) &\overset{I.H.}{=} a\left(a^n c + \frac{a^n - 1}{a-1} b\right) + b \\ &= a^{n+1} c + \frac{a^{n+1} - a}{a-1} b + \frac{a-1}{a-1} b \\ &= a^{n+1} c + \frac{a^{n+1} - 1}{a-1} b \end{aligned}$$

□

The stability of an XOR Arbiter PUF can be determined with a recurrence relation due to its incremental design and the associative property of the XOR. We will use an upper bound on the individual Arbiter PUF stability to obtain a compact term. An exact stability computation is also possible in the same fashion but requires more space.

Theorem 2.19. *Let $\mathring{f}_1, \dots, \mathring{f}_k : \{-1,1\}^n \times \mathbb{R} \to \{-1,1\}$ be noisy Arbiter PUF models as defined in* (2.2.2)*. Let $\mathring{g} : \{-1,1\}^n \times \mathbb{R}^k \to \{-1,1\}$ be the resulting noisy XOR Arbiter PUF defined by*

$$\mathring{g}\left(c, \Delta_{\text{Noise}}\right) = \prod_{i=1}^{k} \mathring{f}_i\left(c, \Delta_{\text{Noise}}^{(i)}\right).$$

Then for a challenge $c \in \{-1,1\}^n$,

$$\text{Stab}_g\left(c\right) \leq \frac{1}{2}\left((2p-1)^k + 1\right),$$

where $p = \max\left\{\text{Stab}_{f_i}\left(c\right) \mid 1 \leq i \leq k\right\}$.

Proof. We show the upper bound of the stability by assuming that all Arbiter PUFs have stability p. Let $P(k-1)$ be the stability of the XOR of the first $k-1$ Arbiter PUFs. Then

$$\begin{aligned} P(k) &= p \cdot P(k-1) + (1-p) \cdot (1 - P(k-1)) \\ &= (2p-1) \cdot P(k-1) + 1 - p \end{aligned}$$

and $P(0) = 1$, since the XORed value remains unchanged if and only if none or both of the inputs flip. We solve this first order non-homogeneous recurrence relation with Lemma 2.18 and get

$$\begin{aligned}
P(k) &= (2p-1)^k + \frac{\left((2p-1)^k - 1\right)}{2p-2}(1-p) \\
&= (2p-1)^k + \frac{(2p-1)^k - p(2p-1)^k + p - 1}{2p-2} \\
&= \frac{(2p-1)^k(2p-2) + (2p-1)^k - p(2p-1)^k + p - 1}{2p-2} \\
&= \frac{2p(2p-1)^k - 2(2p-1)^k + (2p-1)^k - p(2p-1)^k + p - 1}{2p-2} \\
&= \frac{p(2p-1)^k - (2p-1)^k + p - 1}{2p-2} \\
&= \frac{1}{2}\frac{(p-1)\left((2p-1)^k + 1\right)}{p-1} \\
&= \frac{1}{2}\left((2p-1)^k + 1\right).
\end{aligned}$$

Since we assumed at the beginning of the proof that all individual Arbiter PUFs have the maximal stability p, the actual stability is less or equal to this upper bound. □

We can see that the stability of the XOR Arbiter PUF converges exponentially in k to that of a uniformly random function by which the PUF would become unusable. This could only be prevented if the individual Arbiter PUF's stability p would be exponentially close to 1.

2.4 attacks on the (xor) arbiter puf

We remember that the goal of PUFs is to construct a function that cannot be cloned. Since an Arbiter PUF has the same blueprint but different behavior on different silicon implementations as well as an exponentially large challenge space it might seem that therefore it is also secure. Unfortunately, this is not the case. As we saw in the previous section, the Arbiter PUF can be modeled as an LTF. The problem of cloning the Arbiter PUF is hence reduced from building an exact silicon copy to identifying the weight vector w from a set of collected CRPs using ML attacks. For the XOR Arbiter PUF, one needs to identify all k weight vectors of the combined Arbiter PUFs.

2.4.1 *Logistic Regression*

One of the current state-of-the-art attacks is due to Rührmair et. al. using *Logistic Regression* (LR) with resilient propagation (RProp) [Rüh+13].

The attack is applicable to Arbiter PUFs as well as XOR Arbiter PUFs and defines a cost function which is iteratively minimized using gradient descent. The PUF is modeled according to Theorem 2.14 for a single Arbiter PUF and as in (2.3.1) for the XOR Arbiter PUF removing the sgn function. Therefore our model is a function $f : \{-1,1\}^n \times \mathbb{R}^s \to \mathbb{R}$ dependent on a weight vector $\overrightarrow{w}$ with $s = n$ entries for a single Arbiter PUF and $s = n \cdot k$ entries in case of the XOR Arbiter PUF. Since we learn from random examples, we have given a training set L such that $(x, r) \in L$ with input $x \in \{-1,1\}^n$ and response $r \in \{-1,1\}$.

We define a function

$$p\left(x, r, \overrightarrow{w}\right) = \sigma\left(f(x, \overrightarrow{w})r\right) = \left(1 + e^{-f(x,\overrightarrow{w})r}\right)^{-1}$$

that represents the probability that the model f, given the current weights $\overrightarrow{w}$, returns the correct response r for an input x. We define this probability using the sigmoid function σ of the product of our model value and the actual response. If our model returns the correct bit, i.e. $\operatorname{sgn}\left(f(x, \overrightarrow{w})\right) = r$ then $f(x, \overrightarrow{w})r > 0$ and therefore $p\left(x, r\right) \to 1$ as $\left|f(x, \overrightarrow{w})\right| \to \infty$. Alternatively speaking, the certainty of a response increases with the magnitude of the delay difference. This also works the other way around. If $\operatorname{sgn}\left(f(x, \overrightarrow{w})\right) \neq r$ then $f(x, \overrightarrow{w})r < 0$ and hence $p\left(x, r\right) \to 0$ as $\left|f(x, \overrightarrow{w})\right| \to \infty$. Now a high delay difference magnitude means that the model predicts completely in the wrong direction. On the other hand for $\left|f(x, \overrightarrow{w})\right| \to 0$ the probability $p\left(x, r\right) \to 1/2$, since a low delay difference magnitude represents the uncertainty of the model.

The likelihood that the current model is observed under the training set L, is the sum of our probability function for all CRPs in L. We, thus, want to find a $\overrightarrow{w}_{\text{Goal}}$ that maximizes this sum which is equivalent to minimizing the sum of the negative log-likelihood

$$\begin{aligned}\overrightarrow{w}_{\text{Goal}} &= \operatorname{argmax}_{\overrightarrow{w}} \sum_{(x,r)\in L} p\left(x, r \mid \overrightarrow{w}\right) \\ &= \operatorname{argmin}_{\overrightarrow{w}} \sum_{(x,r)\in L} -\ln\left(p\left(x, r \mid \overrightarrow{w}\right)\right).\end{aligned}$$

Since there exists no known analytical solution to this problem, the minimizing $\overrightarrow{w}$ will be determined iteratively using gradient descent. Because of the special properties of the sigmoid function in combination with the logarithm, we can express the gradient of the log-likelihood function dependent on the gradient of our model f,

$$\nabla\left(\sum_{(x,r)\in L} p\left(x, r \mid \overrightarrow{w}\right)\right) = \sum_{(x,r)\in L} r\left(\sigma\left(f\left(x, \overrightarrow{w}\right) r\right) - 1\right) \nabla f\left(x, \overrightarrow{w}\right).$$

The gradient is determined by the partial derivatives of f with respect to the weights:

$$\nabla f\left(x, \vec{w}\right) = \begin{pmatrix} \frac{\partial}{\partial w_1} f\left(x, \vec{w}\right) \\ \vdots \\ \frac{\partial}{\partial w_s} f\left(x, \vec{w}\right) \end{pmatrix}.$$

If we, for instance, look at the XOR Arbiter PUF, we obtain

$$\frac{\partial}{\partial w_a^{(b)}} f\left(x, \vec{w}\right) = \frac{\partial}{\partial w_a^{(b)}} \prod_{l=1}^{k} \left(\sum_{i=1}^{n} \left(w_i^{(l)} x_i \right) \right) = \prod_{\substack{l=1 \\ l \neq a}}^{k} \left(\sum_{i=1}^{n} \left(w_i^{(l)} x_i \right) \right) x_a.$$

This is because of the fact that

$$\frac{\partial}{\partial w_a^{(b)}} \sum_{i=1}^{n} \left(w_i^{(l)} x_i \right) = \begin{cases} x_a & \text{if } i = a, l = b, \\ 0 & \text{else.} \end{cases}$$

The algorithm starts with a random weight vector $\vec{w}$. In each iteration, the gradient is calculated to see if it is (almost) zero and therefore the algorithm has converged to a minimum. If this is not the case, the weight vector $\vec{w}$ is updated with RProp using the gradient. Hence, in each step, the weight vector is shifted towards a minimum of the negative log-likelihood.

Using the empirical results Rührmair et. al. analyzed the runtime as well as the amount of required CRPs for the different tested PUFs. They estimate that the amount of CRPs required for a successful attack on an Arbiter PUF grows with $N_{CRP} = \mathcal{O}\left({}^{n}/_{\varepsilon}\right)$ where ε is the misclassification rate. For the XOR Arbiter PUF the amount grows with $N_{CRP} = \mathcal{O}\left({}^{nk}/_{\varepsilon}\right)$. The overall runtime for the classification of an Arbiter PUF grows with $\mathcal{O}\left({}^{n^2}/_{\varepsilon} \log {}^{n}/_{\varepsilon}\right)$. For the XOR Arbiter PUF, however, an additional problem exists. In contrast to the conventional Arbiter PUF, the algorithm is not guaranteed to find the global minimum in its first trial. A number of restarts is required such that the initial randomly chosen $\vec{w}$ is an attractor of the minimum. In the other cases, the algorithm converges to a local minimum which does not achieve good prediction rates. The number of required restarts grows with $\mathcal{O}\left({}^{(n+1)^k}/_{N_{CRP}}\right)$ and the time required per run grows with $\mathcal{O}\left(n \cdot k \cdot N_{CRP}\right)$. A comparison of the runtimes and required CRPs can be found in Table 2.1.

Although the attack has an exponential runtime in k it is efficiently applicable to XOR Arbiter PUFs. This is because, as stated in Theorem 2.19, the amount of combined Arbiter PUFs that gives a useful XOR Arbiter PUF is limited due to the exponential stability decrease in k. The attack was also performed on CRPs with added random noise, i.e. for some CRPs the response bit was flipped. The authors stated that to achieve similar results as without noise, only a three to four times increase of CRPs was necessary, whereas the per-CRP runtime did not change significantly.

	Arbiter PUF	XOR Arbiter PUF
N_{CRP}	$\mathcal{O}\left(\frac{n}{\varepsilon}\right)$	$\mathcal{O}\left(\frac{nk}{\varepsilon}\right)$
runtime per run	$\mathcal{O}\left(\frac{n^2}{\varepsilon}\log\frac{n}{\varepsilon}\right)$	$\mathcal{O}\left(n \cdot k \cdot N_{CRP}\right)$
# expected runs	0	$\mathcal{O}\left(\frac{(n+1)^k}{N_{CRP}}\right)$

Table 2.1: Comparison of runtime and number of required CRPs for the Logistic Regression attack

2.4.2 *Reliability Based Attack*

Recently Becker proposed a new attack using side-channel information [Bec15]. It uses Evolution Strategies (ES) with Covariance Matrix Adaption (CMA) to optimize the model using a fitness function. In contrast to Rührmair's attack it does not define the fitness function on the distance of the current model outputs and the CRPs, but uses the instability of the PUF as a metric of the approximation quality. The attack requires multiple measurements of the same challenge to find unstable challenges. Uniformly random CRPs are hence not sufficient. As we saw in Corollary 2.17, for a given challenge a small delay difference is equivalent to having a low stability. Becker uses this fact to compare the stability of the model and the given CRPs. To that end, he defines the notion of reliability h_i for a collection of CRPs of the same challenge $c^{(i)}$. It is computed by

$$h_i = \left| \frac{l}{2} - \sum_{j=1}^{l} r_{i,j} \right| , \qquad (2.4.1)$$

where $r_{i,1}, r_{i,2}, ..., r_{i,l} \in \{0,1\}$ are the individual responses for the challenge $c^{(i)}$. The reliability is similar to our definition of stability. It is minimal for $\text{Stab}_f[c^{(i)}] = 1/2$ and maximal for $\text{Stab}_f[c^{(i)}] = 1$. Again the PUF is modeled as in Theorem 2.14 removing the sgn function. For a model f a hypothetical reliability $\widetilde{h}_i$ is calculated by testing if the absolute delay difference of the model is higher or lower than a certain threshold ϵ, i.e.

$$\widetilde{h}_i = \begin{cases} 1 & \text{if } \left|f\left(c^{(i)}\right)\right| > \epsilon, \\ 0 & \text{if } \left|f\left(c^{(i)}\right)\right| < \epsilon. \end{cases}$$

The algorithm starts with a generation of PUF models by generating random weight vectors $\boldsymbol{w}$ and computing the desired reliability $h = h_1, \ldots, h_N$ for N different challenges as described in (2.4.1). In each iteration, the hypothetical reliability $\widetilde{h} = \widetilde{h}_1, \ldots, \widetilde{h}_N$ of all current PUF models is determined followed by the computation of the fitness function, defined by the Pearson correlation coefficient between h and $\widetilde{h}$. All instances with a fitness level below a certain threshold are discarded. The remaining instances are combined and randomly mutated using CMA to form the generation for the next iteration. This is repeated until the accuracy requirements are met.

The algorithm can also be used to attack the XOR Arbiter PUF by running it multiple times, learning one individual Arbiter PUF after the other. Becker argues that if the algorithm tries to learn a single Arbiter PUF the reliability of the remaining XORed PUFs can be interpreted as noise. The algorithm, however, cannot target a specific individual Arbiter PUF, but rather converges randomly to one of the combined PUFs. Therefore after a certain subset of individual PUFs was already found, the probability to find a new one decreases. Additionally, the convergence probability for a given training set is not uniformly distributed among all Arbiter PUFs, further reducing the probability to find all models. Becker proposes to then use, for instance, the Logistic Regression approach to learn the remaining ones. Although he does not provide a detailed runtime analysis, his empirical results show that his algorithm considerably outperforms the conventional LR approach with increasing number of combined Arbiter PUFs.

2.4.3 *Photonic Emission Analysis*

Up to this point, we only looked at ML attacks. In this case, it suffices for the attacker to have random CRPs or query access to the PUF. However, as noted earlier, a PUF should also satisfy an even stronger criterion. If an attacker gets hold of a PUF instance, that is, he has physical possession of it, he still must not be able to clone the device's behavior.

In this context Tajik et. al. presented a powerful semi-invasive attack on Arbiter PUFs [Taj+17]. They used an optimized infrared microscope, directed at the backside of the PUF, to detect photons emitted by the transistors in the stages of the PUF. Based on this, they were able to optically measure the delay values for the upper and lower path. Although they did not use a classical Arbiter PUF, but rather an implementation on a Field Programmable Gate Array (FPGA) using inverters as stages, their attack is extendable to other delay based PUFs. To perform the attack, only $2n + 2$ specific challenges have to be queried from the PUF (n being the input length). However, a repeated measurement of the same challenge is necessary such that the amount of emitted photons is high enough for the analysis. The measured propagation delay values can then be used to create a system of linear equations, whose solution yields the individual stage delay values and hence a model of the PUF.

An important aspect of this attack is that it can also be applied to the XOR Arbiter PUF. The individual Arbiter PUFs can be learned one by one since the propagation delay values can be determined before the XOR. Hence, the runtime of the attack only grows linear in the number of combined PUFs. The amount of required CRPs stays at $2n + 2$ since the same specific challenges are forwarded to all the Arbiter PUFs.

From the attacks presented, this one has by far the lowest runtime and number of required CRPs. However, it comes with the drawback of having to expose the device's backside which could lead to a modification of its behavior. Furthermore, the costs for the required equipment were around 30 000 Euros.

3 INFLUENCE ANALYSIS

In this chapter we look at a quantification concerning the Fourier weights, namely the influence. We apply the concept to Arbiter PUFs as well as the XOR Arbiter PUF to later on interpret the impact of the results regarding their learnability.

3.1 INFLUENCE

First we define the notion of influence for a Boolean function.

Definition 3.1. The *influence* of the ith coordinate is defined as the probability that the output of the function $f : \{-1,1\}^n \to \{-1,1\}$ flips if the input bit at position i is flipped.

$$\mathbf{Inf}_i[f] = \Pr_{\boldsymbol{x}}\left[f(\boldsymbol{x}) \neq f\left(\boldsymbol{x}^{\oplus i}\right)\right],$$

where $x^{\oplus i} = (x_1, \dots, x_{i-1}, -x_i, x_{i+1}, \dots, x_n)$.

In order to relate this to the Fourier expansion we need to define the influence from another point of view.

Definition 3.2. We define the ith discrete derivative operator D_i mapping a function $f : \{-1,1\}^n \to \mathbb{R}$ to the function $\mathrm{D}_i f : \{-1,1\}^n \to \mathbb{R}$ defined by

$$\mathrm{D}_i f(x) = \frac{f\left(x^{(i\mapsto 1)}\right) - f\left(x^{(i\mapsto -1)}\right)}{2},$$

where $x^{(i\mapsto b)} = (x_1, \dots, x_{i-1}, b, x_{i+1}, \dots, x_n)$.

Later we will need to use the linearity of the operator.

Fact 3.3. *For* $f, g : \{-1,1\}^n \to \mathbb{R}$ *holds* $\mathrm{D}_i(f+g) = \mathrm{D}_i f + \mathrm{D}_i g$.

Proof.

$$\begin{aligned}
\mathrm{D}_i(f+g) &= \frac{(f+g)\left(x^{(i\mapsto 1)}\right) - (f+g)\left(x^{(i\mapsto -1)}\right)}{2} \\
&= \frac{f\left(x^{(i\mapsto 1)}\right) + g\left(x^{(i\mapsto 1)}\right) - f\left(x^{(i\mapsto -1)}\right) - g\left(x^{(i\mapsto -1)}\right)}{2} \\
&= \mathrm{D}_i f + \mathrm{D}_i g
\end{aligned}$$

□

Looking at the effect of the operator for a Boolean valued function $f : \{-1,1\}^n \to \{-1,1\}$ we have

$$\mathrm{D}_i f(x) = \begin{cases} 0 & \text{if } f(x) = f\left(x^{\oplus i}\right), \\ \pm 1 & \text{if } f(x) \neq f\left(x^{\oplus i}\right). \end{cases}$$

Hence $\mathrm{D}_i f(x)^2$ is an 0-1-indicator function for $f(x) \neq f\left(x^{\oplus i}\right)$. We therefore can rewrite the definition of the influence from a probability to an expectation of this indicator function, i.e.

$$\mathbf{Inf}_i[f] = \mathop{\mathbf{E}}_{\boldsymbol{x}\sim\{-1,1\}^n}\left[\mathrm{D}_i f(x)^2\right]. \tag{3.1.1}$$

In the next step we want to determine the Fourier expansion of $\mathrm{D}_i f(x)$. To that end we will look at the impact of the operator on the parity functions.

Fact 3.4. *For any $S \subseteq [n]$,*

$$\mathrm{D}_i \chi_S(x) = \begin{cases} \chi_{S\setminus\{i\}}(x) & \textit{if } i \in S, \\ 0 & \textit{if } i \notin S. \end{cases}$$

Proof. First we consider the case where $i \in S$,

$$\begin{aligned}
\mathrm{D}_i \chi_S(x) &= \frac{\chi_S\left(x^{(i\mapsto 1)}\right) - \chi_S\left(x^{(i\mapsto -1)}\right)}{2} \\
&= \frac{\prod_{j\in S}\left(x^{(i\mapsto 1)}\right)_j - \prod_{j\in S}\left(x^{(i\mapsto -1)}\right)_j}{2} \\
&= \frac{1\cdot\prod_{j\in S\setminus\{i\}} x_j - (-1)\cdot\prod_{j\in S\setminus\{i\}} x_j}{2} \\
&= \chi_{S\setminus\{i\}}(x).
\end{aligned}$$

For $i \notin S$ in the third equality we get

$$\begin{aligned}
\mathrm{D}_i \chi_S(x) &= \frac{\prod_{j\in S} x_j - \prod_{j\in S} x_j}{2} \\
&= 0.
\end{aligned}$$

□

Now we are able to determine the Fourier expansion of $\mathrm{D}_i f(x)$ for any f.

Proposition 3.5. *Let $f : \{-1,1\}^n \to \mathbb{R}$ have the Fourier expansion $f(x) = \sum_{S\subseteq[n]} \widehat{f}(S)\,\chi_S(x)$. Then*

$$\mathrm{D}_i f(x) = \sum_{\substack{S\subseteq[n] \\ S\ni i}} \widehat{f}(S)\,\chi_{S\setminus\{i\}}(x).$$

Proof. Using Fact 3.3 we get

$$\mathrm{D}_i f(x) = \sum_{S \subseteq [n]} \widehat{f}(S)\, \mathrm{D}_i \chi_S(x).$$

By Fact 3.4 the claim follows. □

Now we are finally able to create the connection between the influence and the Fourier coefficients.

Theorem 3.6. *Let $f : \{-1,1\}^n \to \{-1,1\}$, then for a fixed $i \in [n]$ we have*

$$\mathbf{Inf}_i[f] = \sum_{\substack{S \subseteq [n] \\ S \ni i}} \widehat{f}(S)^2 .$$

Proof. We only need to combine everything we have pointed out so far.

$$\begin{aligned}
\mathbf{Inf}_i[f] &= \mathop{\mathbf{E}}_{\boldsymbol{x} \sim \{-1,1\}^n} \left[\mathrm{D}_i f(x)^2\right] && \text{(From (3.1.1))} \\
&= \sum_{S \subseteq [n]} \widehat{\mathrm{D}_i f}(S)^2 && \text{(Theorem 2.5)} \\
&= \sum_{\substack{S \subseteq [n] \\ i \in S}} \widehat{f}(S)^2 && \text{(Proposition 3.5)}
\end{aligned}$$

□

The influence can be extended to the notion of the total influence, also referred to as the average sensitivity.

Definition 3.7. We define the total influence $\mathbf{I}[f]$ of a function $f : \{-1,1\}^n \to \{-1,1\}$ as the sum of its individual influences

$$\mathbf{I}[f] = \sum_{i=1}^{n} \mathbf{Inf}_i[f].$$

The total influence also has a connection to the Fourier coefficients. Since we know about the individual influence we can easily extend this to the total influence.

Theorem 3.8. *Let $f : \{-1,1\}^n \to \{-1,1\}$, then*

$$\mathbf{I}[f] = \sum_{S \subseteq [n]} |S|\, \widehat{f}(S)^2 .$$

Proof. First we use Definition 3.7 and Theorem 3.6 and get

$$\begin{aligned}
\mathbf{I}[f] &= \sum_{i=1}^{n} \mathbf{Inf}_i[f] \\
&= \sum_{i=1}^{n} \sum_{\substack{S \subseteq [n] \\ i \in S}} \widehat{f}(S)^2 .
\end{aligned}$$

We observe that for any given S, the coefficient $\widehat{f}(S)^2$ appears once for each $i \in S$ in the sum, hence

$$\mathbf{I}[f] = \sum_{S \subseteq [n]} |S| \widehat{f}(S)^2 .$$

$\square$

3.2 APPLICATION ON ARBITER PUFS

As we now have introduced the notion of the (total) influence, we want to apply it on Arbiter PUFs. Since we are not looking at a single mathematical function, but rather at the general concept of Arbiter PUFs, it is pointless to determine the influence of a single PUF instance. Instead we will assume that the delay impacts of the stages are independently Gaussian distributed, since they are based on physical imperfections [**Lim2005-eo**]. We will determine expected values and probability distributions building on the distribution of the weights.

3.2.1 *Influence of Arbiter PUFs*

The following theorem, determining the expected influence of an Arbiter PUF, was first shown by Hammouri et. al. [HS08].

Theorem 3.9. *Let $f : \{-1,1\}^n \to \{-1,1\}$ be an ATF where the weights w_i are drawn independently from the same Gaussian distribution with zero mean. Then*

$$\mathop{\mathbf{E}}_{\boldsymbol{w}_i}[\mathbf{Inf}_z[f]] = \frac{2}{\pi} \arctan\left(\sqrt{\frac{z}{n+1-z}}\right).$$

Proof. The PUF is modeled as $f : \{-1,1\}^n \to \{-1,1\}$, with

$$f(c) = \operatorname{sgn}\left(w_0 + \sum_{i=1}^{n}\left(w_i \prod_{j=i}^{n} c_j\right)\right).$$

We want to calculate the expected value of the influence of the zth bit under the condition that the individual weights $\boldsymbol{w}_0, ..., \boldsymbol{w}_n$ are chosen independently from the same Gaussian distribution with zero mean:

$$\mathop{\mathbf{E}}_{\boldsymbol{w}_i \sim \mathcal{N}(0,\sigma_w^2)}[\mathbf{Inf}_z[f]] = \mathop{\mathbf{E}}_{\boldsymbol{w}_i \sim \mathcal{N}(0,\sigma_w^2)}\left[\mathop{\mathbf{Pr}}_{\boldsymbol{c} \sim \{-1,1\}^n}\left[f(\boldsymbol{c}) \neq f\left(\boldsymbol{c}^{\oplus z}\right)\right]\right],$$

where $\boldsymbol{c}^{\oplus z}$ means that the zth bit in $\boldsymbol{c}$ is flipped. We can rewrite the probability as the expected value of a 0-1-indicator:

$$\mathop{\mathbf{E}}_{\boldsymbol{w}_i \sim \mathcal{N}(0,\sigma_w^2)}[\mathbf{Inf}_z[f]] = \mathop{\mathbf{E}}_{\boldsymbol{w}_i \sim \mathcal{N}(0,\sigma_w^2)}\left[\mathop{\mathbf{E}}_{\boldsymbol{c} \sim \{-1,1\}^n}\left[\frac{1}{2} - \frac{1}{2} f(\boldsymbol{c}) f\left(\boldsymbol{c}^{\oplus z}\right)\right]\right].$$

Since $\boldsymbol{c}$ is drawn out of a discrete distribution we can write the expected value as a scaled sum:

$$\begin{aligned}\mathbf{Inf}_z[f] &= \mathop{\mathbf{E}}_{\boldsymbol{w}_i}\left[\frac{1}{2^n}\sum_{c\in\{-1,1\}^n}\left(\frac{1}{2}-\frac{1}{2}f(c)f\left(c^{\oplus z}\right)\right)\right]\\ &\overset{\text{lin.}}{=} \frac{1}{2^n}\sum_{c\in\{-1,1\}^n}\mathop{\mathbf{E}}_{\boldsymbol{w}_i}\left[\frac{1}{2}-\frac{1}{2}f(c)f\left(c^{\oplus z}\right)\right]\\ &= \frac{1}{2^n}\sum_{c\in\{-1,1\}^n}\mathop{\mathbf{Pr}}_{\boldsymbol{w}_i}\left[f(c)\neq f\left(c^{\oplus z}\right)\right].\end{aligned}$$

In order to calculate $\mathbf{Pr}_{\boldsymbol{w}_i}\left[f(c)\neq f\left(c^{\oplus z}\right)\right]$ for a fixed challenge c we split the sum in f in two parts:

$$f(c)=\operatorname{sgn}\left(\boldsymbol{W}_H(c)+\boldsymbol{W}_T(c)\right),$$

where

$$\boldsymbol{W}_H(c)=\sum_{i=1}^{z}\left(\boldsymbol{w}_i\prod_{j=i}^{n}c_j\right)\text{ and }\boldsymbol{W}_T(c)=\boldsymbol{w}_0+\sum_{i=z+1}^{n}\left(\boldsymbol{w}_i\prod_{j=i}^{n}c_j\right).$$

We observe that

$$\boldsymbol{W}_H(c)=-\boldsymbol{W}_H(c^{\oplus z})\text{ and }\boldsymbol{W}_T(c)=\boldsymbol{W}_T(c^{\oplus z}),$$

since c_z is contained in every term in $\boldsymbol{W}_H(c)$ but in none of $\boldsymbol{W}_T(c)$. Now, considering an arbitrary c from the sum we can calculate the probability. We assume $\boldsymbol{W}_H(c)\neq 0$ since an exact addition of the delay values to 0 is physically impossible.

$$\begin{aligned}\mathop{\mathbf{Pr}}_{\boldsymbol{w}_i}\left[f(c)\neq f\left(c^{\oplus z}\right)\right] &= \mathop{\mathbf{Pr}}_{\boldsymbol{w}_i}\left[\operatorname{sgn}\left(\boldsymbol{W}_H(c)+\boldsymbol{W}_T(c)\right)\neq\operatorname{sgn}\left(\boldsymbol{W}_H(c^{\oplus z})+\boldsymbol{W}_T(c^{\oplus z})\right)\right]\\ &= \mathop{\mathbf{Pr}}_{\boldsymbol{w}_i}\left[\operatorname{sgn}\left(\boldsymbol{W}_H(c)+\boldsymbol{W}_T(c)\right)\neq\operatorname{sgn}\left(\boldsymbol{W}_T(c)-\boldsymbol{W}_H(c)\right)\right]\\ &= \mathop{\mathbf{Pr}}_{\boldsymbol{w}_i}\left[\operatorname{sgn}\left(1+\frac{\boldsymbol{W}_T(c)}{\boldsymbol{W}_H(c)}\right)\neq\operatorname{sgn}\left(\frac{\boldsymbol{W}_T(c)}{\boldsymbol{W}_H(c)}-1\right)\right]\\ &= \mathop{\mathbf{Pr}}_{\boldsymbol{w}_i}\left[1+\frac{\boldsymbol{W}_T(c)}{\boldsymbol{W}_H(c)}>0\wedge\frac{\boldsymbol{W}_T(c)}{\boldsymbol{W}_H(c)}-1<0\right]\\ &\quad+\mathop{\mathbf{Pr}}_{\boldsymbol{w}_i}\left[1+\frac{\boldsymbol{W}_T(c)}{\boldsymbol{W}_H(c)}<0\wedge\frac{\boldsymbol{W}_T(c)}{\boldsymbol{W}_H(c)}-1>0\right]\\ &= \mathop{\mathbf{Pr}}_{\boldsymbol{w}_i}\left[-1<\frac{\boldsymbol{W}_T(c)}{\boldsymbol{W}_H(c)}<1\right]+\mathop{\mathbf{Pr}}_{\boldsymbol{w}_i}\left[\frac{\boldsymbol{W}_T(c)}{\boldsymbol{W}_H(c)}<-1\wedge\frac{\boldsymbol{W}_T(c)}{\boldsymbol{W}_H(c)}>1\right]\\ &= \mathop{\mathbf{Pr}}_{\boldsymbol{w}_i}\left[-1<\frac{\boldsymbol{W}_T(c)}{\boldsymbol{W}_H(c)}<1\right]\end{aligned}$$

The random variables $\boldsymbol{W}_H(c)$ and $\boldsymbol{W}_T(c)$ are formed by adding or subtracting the normal distributed random variables $\boldsymbol{w}_i$. If we add or subtract normal distributed random variables the result is also normal distributed. The mean of the new distribution is formed by addition or subtraction of the individual means respectively. The new variance however is always the sum of the individual variances. All our means are 0, so we can calculate the variance by counting the number of $\boldsymbol{w}_i$ contained in the sums.

$$\begin{aligned} \boldsymbol{W}_H(c) &\sim \mathcal{N}\left(0, z\sigma_w^2\right) \\ \boldsymbol{W}_T(c) &\sim \mathcal{N}\left(0, (n+1-z)\,\sigma_w^2\right) \end{aligned}$$

The fraction of two independent and normal distributed random variables with zero mean is Cauchy distributed. The scale value γ is determined by the fraction of the standard deviations of the normal distributed random variables, hence

$$\frac{\boldsymbol{W}_T(c)}{\boldsymbol{W}_H(c)} \sim \text{Cauchy}\left(t = 0, \gamma = \sqrt{\frac{z}{n+1-z}}\right).$$

Finally, knowing the cumulative density function (CDF) Φ of the Cauchy distribution with

$$\Phi(x) = \frac{1}{2} + \frac{1}{\pi}\arctan\left(\frac{x-t}{\gamma}\right)$$

we can compute the probability

$$\begin{aligned} \underset{\boldsymbol{w}_i}{\mathbf{Pr}}\left[-1 < \frac{\boldsymbol{W}_T(c)}{\boldsymbol{W}_H(c)} < 1\right] &= \Phi(1) - \Phi(-1) \\ &= \frac{2}{\pi}\arctan\left(\sqrt{\frac{z}{n+1-z}}\right). \end{aligned}$$

Applying this result yields

$$\begin{aligned} \underset{\boldsymbol{w}_i}{\mathbf{E}}\left[\mathbf{Inf}_z[f]\right] &= \frac{1}{2^n}\sum_{c\in\{-1,1\}^n} \underset{\boldsymbol{w}_i}{\mathbf{Pr}}\left[f(c) \neq f\left(c^{\oplus z}\right)\right] \\ &= \frac{1}{2^n}\sum_{c\in\{-1,1\}^n} \frac{2}{\pi}\arctan\left(\sqrt{\frac{z}{n+1-z}}\right) \\ &= \frac{2}{\pi}\arctan\left(\sqrt{\frac{z}{n+1-z}}\right). \end{aligned}$$

□

Since we now have the individual influence of each bit we can determine the total influence.

Theorem 3.10. *Let $f : \{-1,1\}^n \to \{-1,1\}$ be an ATF where the weights w_i are drawn independently from the same Gaussian distribution with zero mean. Then*

$$\underset{\boldsymbol{w}_i}{\mathbf{E}}\left[\mathbf{I}[f]\right] = \frac{n}{2}.$$

Proof. Using Theorem 3.9 and Definition 3.7 we get

$$\begin{aligned}
\mathop{\mathbf{E}}_{\boldsymbol{w}_i}[\mathbf{I}[f]] &= \mathop{\mathbf{E}}_{\boldsymbol{w}_i}\left[\sum_{z=1}^{n}\mathbf{Inf}_z[f]\right] \\
&\overset{\text{lin.}}{=} \sum_{z=1}^{n}\mathop{\mathbf{E}}_{\boldsymbol{w}_i}[\mathbf{Inf}_z[f]] \\
&= \sum_{z=1}^{n}\frac{2}{\pi}\arctan\left(\sqrt{\frac{z}{n+1-z}}\right).
\end{aligned}$$

First we show that the statement holds for even n. We will use the property $\arctan(x) = \frac{\pi}{2} - \arctan\frac{1}{x}$ since our x is always positive.

$$\begin{aligned}
\mathop{\mathbf{E}}_{\boldsymbol{w}_i}[\mathbf{I}[f]] &= \sum_{z=1}^{n}\frac{2}{\pi}\arctan\left(\sqrt{\frac{z}{n+1-z}}\right) \\
&= \sum_{z=1}^{n/2}\frac{2}{\pi}\arctan\left(\sqrt{\frac{z}{n+1-z}}\right) + \sum_{z=n/2+1}^{n}\frac{2}{\pi}\arctan\left(\sqrt{\frac{z}{n+1-z}}\right) \\
&= \sum_{z=1}^{n/2}\frac{2}{\pi}\arctan\left(\sqrt{\frac{z}{n+1-z}}\right) + \sum_{z=n/2+1}^{n}1-\frac{2}{\pi}\arctan\left(\sqrt{\frac{n+1-z}{z}}\right) \\
&= \frac{n}{2} + \sum_{z=1}^{n/2}\frac{2}{\pi}\arctan\left(\sqrt{\frac{z}{n+1-z}}\right) - \sum_{z=n/2+1}^{n}\frac{2}{\pi}\arctan\left(\sqrt{\frac{n+1-z}{z}}\right)
\end{aligned}$$

If we reverse one sum the addends of both sums are pairwise equal. We can verify this by substituting $z' = n - z + 1$ in the second sum. For odd n the verification is similar. We split the sum in the middle, left, and right part and use the same proof method. □

We now have quantified the expected value for the influence and total influence of an Arbiter PUF. Another interesting question is to describe the probability distribution of the influence. This can be used to determine, for instance, the variance.

Theorem 3.11. *Let $f : \{-1,1\}^n \to \{-1,1\}$ be an unbiased ATF, then*

$$\mathop{\mathbf{Pr}}_{\boldsymbol{w}_i}[\mathbf{Inf}_z[f] < x] \approx F\left(\tan^2\left(\frac{\pi}{2}x\right) \mid z; n-z\right),$$

with $F(x \mid d_1; d_2)$ being the CDF of the F-distribution with d_1 degrees of freedom in the numerator and d_2 in the denominator.

Proof. The PUF is modeled as $f : \{-1,1\}^n \to \{-1,1\}$, with

$$f(c) = \operatorname{sgn}\left(\sum_{i=1}^{n}\left(w_i\prod_{j=i}^{n}c_j\right)\right).$$

We can assume without loss of generality that $\sigma_w^2 = 1$, since in an LTF if all weights are scaled by the same positive factor the function does not change. This time we want to know the probability distribution of the influence of the zth input bit by computing its CDF, i.e.

$$\Pr_{\boldsymbol{w}_i \sim \mathcal{N}(0,\sigma_w^2)} [\mathbf{Inf}_z [f] < x] = \Pr_{\boldsymbol{w}_i \sim \mathcal{N}(0,\sigma_w^2)} \left[\Pr_{\boldsymbol{c} \sim \{-1,1\}^n} \left[f(\boldsymbol{c}) \neq f\left(\boldsymbol{c}^{\oplus z}\right) \right] < x \right].$$

We use the same transformation as in the proof of Theorem 3.9 that

$$\mathbf{Pr}\left[f(\boldsymbol{c}) \neq f\left(\boldsymbol{c}^{\oplus z}\right) \right] = \mathbf{Pr}\left[-1 < \frac{\boldsymbol{W}_T(\boldsymbol{c})}{\boldsymbol{W}_H(\boldsymbol{c})} < 1 \right].$$

It does not matter that this time we have a probability with the challenge being the random variable instead of the weights. The transformation is applicable nonetheless. We hence get

$$\Pr_{\boldsymbol{w}_i} [\mathbf{Inf}_z [f] < x] = \Pr_{\boldsymbol{w}_i} \left[\Pr_{\boldsymbol{c}} \left[-1 < \frac{\boldsymbol{W}_T(\boldsymbol{c})}{\boldsymbol{W}_H(\boldsymbol{c})} < 1 \right] < x \right].$$

In order to apply the Berry-Esseen Theorem (using the O'Donnel formulation) we use Theorem 2.14 to define the alternative input vector $x = (x_1, \ldots, x_n)$ with $x_i = \prod_{j=i}^{n} c_j$. Since this mapping is onto one-to-one we can consider $w_i \boldsymbol{x_i}$ random variables with $\mathbf{E}[w_i \boldsymbol{x_i}] = 0$ and $\mathbf{Var}[w_i \boldsymbol{x_i}] = w_i^2$. Now, approximating $\boldsymbol{W}_T(\boldsymbol{c})$ and $\boldsymbol{W}_H(\boldsymbol{c})$ individually with Gaussian distributions we get the following errors

$$\begin{aligned} |\mathbf{Pr}[\boldsymbol{W}_H(\boldsymbol{c}) \leq u] - \mathbf{Pr}[\boldsymbol{Z}_H \leq u]| &\leq c \left(\sum_{i=1}^{z} \mathbf{Var}[w_i \boldsymbol{x_i}] \right)^{-\frac{3}{2}} \left(\sum_{i=1}^{z} \mathbf{E}\left[|w_i \boldsymbol{x}_i - \mathbf{E}[w_i \boldsymbol{x}_i]|^3 \right] \right) \\ &= c \left(\sum_{i=1}^{z} w_i^2 \right)^{-\frac{3}{2}} \left(\sum_{i=1}^{z} \mathbf{E}\left[|w_i \boldsymbol{x}_i|^3 \right] \right) \\ &= c \left(\sum_{i=1}^{z} w_i^2 \right)^{-\frac{3}{2}} \left(\sum_{i=1}^{z} w_i^3 \right), \end{aligned}$$

where $\boldsymbol{Z}_H \sim \mathcal{N}\left(0, \left(\sum_{i=1}^{z} w_i^2\right)\right)$ as well as

$$|\mathbf{Pr}[\boldsymbol{W}_T(\boldsymbol{c}) \leq u] - \mathbf{Pr}[\boldsymbol{Z}_T \leq u]| \leq c \left(\sum_{i=z+1}^{n} w_i^2 \right)^{-\frac{3}{2}} \left(\sum_{i=z+1}^{n} w_i^3 \right),$$

where $\boldsymbol{Z}_T \sim \mathcal{N}\left(0, \left(\sum_{i=z+1}^{n} w_i^2\right)\right)$ and $c = .56$ is a universal constant. Using this approximation we have a fraction of two normal distributed variables, hence

$$\begin{aligned} \Pr_{\boldsymbol{w}_i} [\mathbf{Inf}_z [f] < x] &= \Pr_{\boldsymbol{w}_i} \left[\Pr_{\boldsymbol{c}} \left[-1 < \frac{\boldsymbol{W}_T(\boldsymbol{c})}{\boldsymbol{W}_H(\boldsymbol{c})} < 1 \right] < x \right] \\ &\approx \Pr_{\boldsymbol{w}_i} \left[\Pr_{\boldsymbol{c}} \left[-1 < \frac{\boldsymbol{Z}_T(\boldsymbol{c})}{\boldsymbol{Z}_H(\boldsymbol{c})} < 1 \right] < x \right] \\ &= \Pr_{\boldsymbol{w}_i} \left[\frac{2}{\pi} \arctan\left(\sqrt{\frac{\sum_{i=1}^{z} \boldsymbol{w}_i^2}{\sum_{i=z+1}^{n} \boldsymbol{w}_i^2}} \right) < x \right] \end{aligned}$$

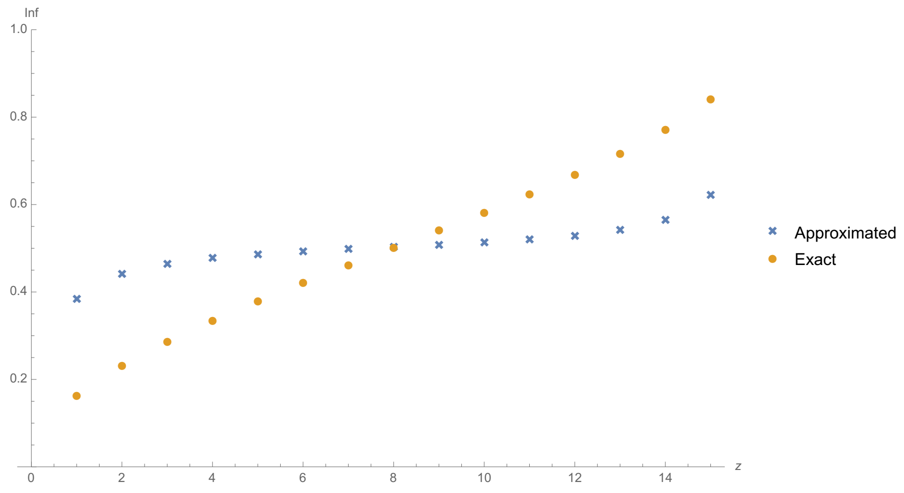

Figure 3.2.1: Comparison of expected influences ($n = 16$) for the exact values as in Theorem 3.9 and the numerically determined expectations for Theorem 3.11

To calculate the probability we rearrange for the weights

$$\Pr_{\boldsymbol{w}_i}[\mathbf{Inf}_z[f] < x] = \Pr_{\boldsymbol{w}_i}\left[\frac{\sum_{i=1}^{z}\boldsymbol{w}_i^2}{\sum_{i=z+1}^{n}\boldsymbol{w}_i^2} < \tan^2\left(\frac{\pi}{2}x\right)\right].$$

Now we have a fraction of sums of squared standard normal distributed variables (since $\sigma_w^2 = 1$) which is F-distributed. The nominator as z degrees of freedom and the denominator has $n - z$. $\square$

To compare the approximated results from Theorem 3.11 with the exact ones from Theorem 3.9, we numerically determined $\mathbf{E}[\mathbf{Inf}_z[f]] = \mathbf{E}\left[2/\pi \arctan\left(\sqrt{\boldsymbol{x}}\right)\right]$ for $z \in \{1, \dots, n-1\}$, where $\boldsymbol{x} \sim F(z; n-z)$. All numerical estimations where computed with Wolfram Mathematica 11. As displayed in Figure 3.2.1 we can see that the expectations do not match exactly. This is most likely due to the error introduced by the use of the Berry-Esseen-Theorem. However the general shape of the curve is similar, the first bits have low influence whereas the last bits have higher influence.

Knowing the probability distribution we are now also able to numerically estimate the variance of the influence by calculating $\mathbf{Var}[\boldsymbol{x}] = \mathbf{E}\left[(\boldsymbol{x} - \mathbf{E}[\boldsymbol{x}])^2\right]$. As can be seen in Figure 3.2.2 the variance is higher for the first and last bits and lower for the middle ones. Additionally, a decrease in variance for increasing n can be recognized. However the rate of decrease is slower for the first and last bits, and faster for the bits in the middle.

3.2.2 *Influence of XOR Arbiter PUFs*

Knowing the expected influence of a single Arbiter PUF we can extend the result to the XOR Arbiter PUF.

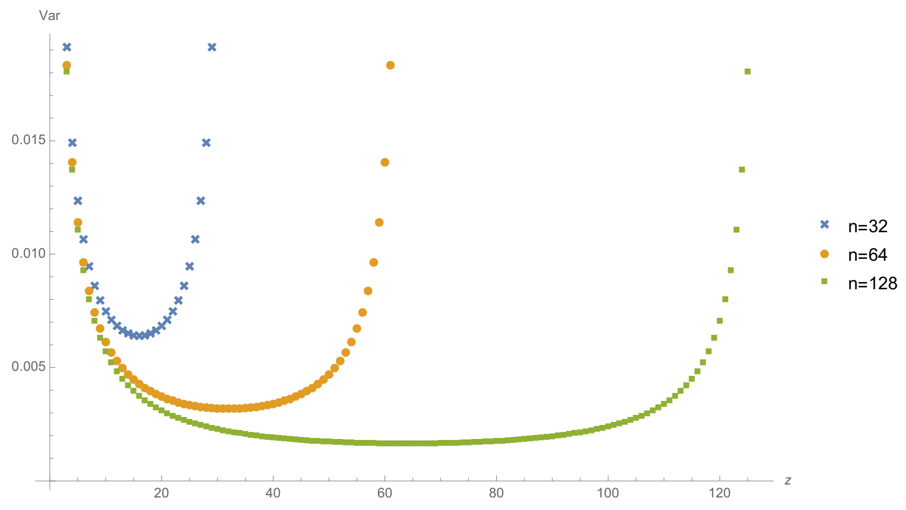

Figure 3.2.2: Variance of the influence of the zth input bit for different n using the distribution from Theorem 3.11

Theorem 3.12. *Let $f_1, \ldots, f_k : \{-1,1\}^n \to \{-1,1\}$ be ATFs where the weights w_i are drawn independently from the same Gaussian distribution with zero mean. Let further $g^{(k)} : \{-1,1\}^n \to \{-1,1\}$ with $g^{(k)}(x) = \prod_{i=1}^{k} f_i(x)$ be an XOR Arbiter PUF. Then*

$$\mathop{\mathbf{E}}_{\boldsymbol{w}_i}\left[\mathbf{Inf}_z\left[g^{(k)}\right]\right] = \frac{1}{2}\left(1-(1-2p)^k\right)$$

with $p = \mathbf{E}_{\boldsymbol{w}_i}[\mathbf{Inf}_z[f_1]] = \ldots = \mathbf{E}_{\boldsymbol{w}_i}[\mathbf{Inf}_z[f_k]]$.

Proof. The proof is similar to that of Theorem 2.19. Again we use a recursive approach to determine the influence of the XOR Arbiter PUF g. The influence is defined as $\mathbf{Inf}_z[g] = \mathbf{Pr}_{\boldsymbol{x}}[g(\boldsymbol{x}) \neq g(\boldsymbol{x}^{\oplus z})]$, therefore we determine in which cases the output bit of g flips. As a shorthand, let

$$P(k-1) = \mathop{\mathbf{E}}_{\boldsymbol{w}_i}\left[\mathbf{Inf}_z\left[g^{(k-1)}\right]\right] \text{ and } p = \mathop{\mathbf{E}}_{\boldsymbol{w}_i}\left[\mathbf{Inf}_z[f_k]\right],$$

then using the independence of $\mathbf{Inf}_z\left[g^{(k-1)}\right]$ and $\mathbf{Inf}_z[f_k]$ we get

$$P(k) = P(k-1)(1-p) + (1-P(k-1))\,p = (1-2p)P(k-1) + p$$

since the output of a 2 bit input XOR flips if exactly one of the inputs flips. Additionally we set $P(0) = 0$. Again we have a first-order non-homogeneous recurrence relation. Applying Lemma 2.18 yields

$$\begin{aligned} P(k) &= \frac{(1-2p)^k - 1}{1-2p-1}p \\ &= \frac{(1-2p)^k - 1}{-2p}p \\ &= \frac{1}{2}\left(1-(1-2p)^k\right). \end{aligned}$$

□

We can see that for large k, the influence of the XOR Arbiter PUF converges to $1/2$. What is also interesting is that this result does not only apply for XOR combined Arbiter PUFs. As long as the combined functions $f_1, \ldots, f_k$ have the same (expected) influence on the same bit, the (expected) influence of the XOR combination of these functions converges to $1/2$. We can also construct a special case of the XOR Arbiter PUF to have an expected influence of exactly $1/2$. We choose $k = n$ odd and pass the original input to every individual PUF with a distinct permutation. These permutations have to satisfy that an input bit c_i is passed to a unique position j for each individual Arbiter PUF. Hence for any two target positions j, j' of an input bit c_i must hold $j \neq j'$.

Theorem 3.13. *Let $f_1, \ldots, f_n : \{-1,1\}^n \to \{-1,1\}$ be ATFs where the weights w_i are drawn independently from the same Gaussian distribution with zero mean. We define permutations $\pi_1, \ldots, \pi_n : [n] \to [n]$ that for each $j \in [n]$ satisfy $\bigcup_{i=1}^{n} \{\pi_i(j)\} = [n]$. Let $g : \{-1,1\}^n \to \{-1,1\}$ be defined by*

$$g^{(n)}(x) = \operatorname{sgn}\left(\prod_{i=1}^{n} f\left(x_{\pi_i(1)}, \ldots, x_{\pi_i(n)}\right)\right),$$

then for odd n and $z \in [n]$ we have $\mathbf{E}_{\boldsymbol{w}_i}\left[\mathbf{Inf}_z\left[g^{(n)}\right]\right] = 1/2$.

Proof. We will again use an recursive approach to prove this theorem. Thus as a shorthand for an arbitrary but fixed $z \in [n]$ we define $P(n) = \mathbf{E}_{\boldsymbol{w}_i}\left[\mathbf{Inf}_z\left[g^{(n)}\right]\right]$. Let $\hat{\imath}$ be the unique index with $\pi_{\hat{\imath}}(z) = (n+1)/2$. It exists since by assumption $\bigcup_{i=1}^{n} \{\pi_i(z)\} = [n]$ and is unique because we have exactly n permutations that have to attain n different values $j \in [n]$. As our basis we choose

$$\begin{aligned} P(1) &= \mathop{\mathbf{E}}_{\boldsymbol{w}_i}\left[\mathbf{Inf}_{(n+1)/2}\left[f_{\hat{\imath}}\right]\right] \\ &= \frac{2}{\pi}\arctan\left(\sqrt{\frac{(n+1)/2}{n+1-(n+1)/2}}\right) && \text{(Theorem 3.9)} \\ &= \frac{1}{2}. \end{aligned}$$

Our recursive proof builds on the fact already used in the proof of Theorem 3.10, i.e. for an $i \in [n]$ and an ATF f,

$$\mathop{\mathbf{E}}_{\boldsymbol{w}_i}\left[\mathbf{Inf}_i[f]\right] + \mathop{\mathbf{E}}_{\boldsymbol{w}_i}\left[\mathbf{Inf}_{(n+1)-i}[f]\right] = 1.$$

As a shorthand we write those pairs of expected influences as p_1, p_2. Since z is mapped uniquely for each individual PUF we have exactly $(n-1)/2$ of those pairs. In every recursive step we take one pair and determine the resulting influence. Since the XOR flips if an odd number of inputs flip and the influences of the individual PUFs are independent from each other we write

$$\begin{aligned} P(n) = {} & p_1 (1 - p_2) (1 - P(n-2)) \\ & + (1 - p_1) p_2 (1 - P(n-2)) \\ & + (1 - p_1)(1 - p_2) P(n-2) \\ & + p_1 p_2 P(n-2). \end{aligned}$$

Using the assumption that $p_1 + p_2 = 1$ we obtain

$$P(n) = (4p_1p_2 - 1) P(n-2) - 2p_1p_2 + 1.$$

Now we can prove by induction on n that $P(n) = \mathbf{E}_{\boldsymbol{w}_i}\left[\mathbf{Inf}_z\left[g^{(n)}\right]\right] = 1/2$ for an arbitrary but fixed $z \in [n]$.
Basis: The statement holds for $n = 1$ by definition, since $P(1) = 1/2$.
Inductive Step: If the statement holds for $n - 2$ it also holds for n.

$$\begin{aligned} P(n) &= (4p_1p_2 - 1) P(n-2) - 2p_1p_2 + 1 \\ &\overset{I.H.}{=} (4p_1p_2 - 1)\frac{1}{2} - 2p_1p_2 + 1 \\ &= \frac{1}{2} \end{aligned}$$

□

We hence can say that this special construction expectably satisfies the *strict avalanche criterion* (SAC). A cryptographic function satisfies the SAC if each output bit flips with probability one half whenever a single input bit is inverted [WT86]. Since we have only one output bit the prerequisites are met. The SAC has evolved to an important measure for cryptography since it formalizes the unpredictability of a cryptographic function. However, if a cryptographic function satisfies the SAC it does not necessarily mean that it is secure.

3.3 PROBABLY APPROXIMATELY CORRECT LEARNING

In this section we will use the previous results to deduce bounds on runtime needed for learning Arbiter PUFs. Therefore we introduce the notion of PAC learning and how it relates to the Fourier expansion.

The name *Probably Approximately Correct* (PAC) learning is a generic term for a class of probabilistic algorithms that satisfy certain requirements. The concept was first introduced by Valiant [Val84]. A PAC learning algorithm takes two parameters $\varepsilon \in [0, 1/2)$ and $\delta \in [0, 1)$, and a function $f : \{-1,1\}^n \to \{-1,1\}$ with either

1. random example access, obtaining m CRPs $(x, f(x))$ with all x being independently drawn uniformly at random; or

2. query access, where the algorithm can repeatedly and adaptively choose an arbitrary challenge x and query the corresponding output $f(x)$.

The algorithm, with probability $1-\delta$, returns a model $g:\{-1,1\}^n \to \{-1,1\}$ which is ε-close to the target function f, i.e. it satisfies $\operatorname{dist}(f,g) \leq \varepsilon$. In contrast to other ML algorithms the user can exactly specify how precise the result should, usually opening the option for a trade-off of accuracy and runtime.

3.3.1 *The Kushilevitz-Mansour Algorithm*

In order to relate PAC learning and the Fourier expansion we introduce the term of ε-concentration.

Definition 3.14. Let $\mathscr{F}$ be a collection of subsets $S \subseteq [n]$. We say that the Fourier spectrum of $f:\{-1,1\}^n \to \{-1,1\}$ is ε-concentrated on $\mathscr{F}$ if

$$\sum_{\substack{S\subseteq[n]\\ S\notin\mathscr{F}}} \widehat{f}(S)^2 \leq \varepsilon.$$

Additionally we say that the Fourier spectrum of f is ε-concentrated up to degree k if

$$\sum_{\substack{S\subseteq[n]\\ |S|>k}} \widehat{f}(S)^2 \leq \varepsilon.$$

We can see that the concentration means that the Fourier weight is not distributed equally among all coefficients but focuses on specific sets. This in turn means that the function f is mostly determined by the Fourier coefficients in this collection. One way of PAC learning is thus to approximate these coefficients using random examples to create a hypothesis $h:\{-1,1\}^n \to \{-1,1\}$ that is ε-close to the original function. This approach was introduced by Mansour in 1994 [Man94]. We first prove the following observation.

Proposition 3.15. *Let $f:\{-1,1\}^n \to \{-1,1\}$ and $g:\{-1,1\}^n \to \mathbb{R}$. Additionally let $h:\{-1,1\}^n \to \{-1,1\}$ be defined as $h(x)=\operatorname{sgn}(g(x))$. Then*

$$\operatorname{dist}(f,h) \leq \mathop{\mathbf{E}}_{\boldsymbol{x}}\left[(f(\boldsymbol{x})-g(\boldsymbol{x}))^2\right].$$

Proof. We use the fact that $(f(x)-g(x))^2 \geq 1$ if $f(x) \neq h(x)$, since then $|f(x)-g(x)| \geq 1$. Then

$$\operatorname{dist}(f,h) = \mathop{\mathbf{Pr}}_{\boldsymbol{x}}[f(\boldsymbol{x}) \neq h(\boldsymbol{x})] = \mathop{\mathbf{E}}_{\boldsymbol{x}}\left[1_{f(\boldsymbol{x})\neq h(\boldsymbol{x})}\right] \leq \mathop{\mathbf{E}}_{\boldsymbol{x}}\left[(f(\boldsymbol{x})-g(\boldsymbol{x}))^2\right]$$

□

```
Input: $m$ CRPs $\left(x^{(i)}, f\left(x^{(i)}\right)\right)$, Collection $\mathscr{F}$
foreach $S \in \mathscr{F}$ do
    Compute $\widetilde{f}(S) = \frac{1}{m}\sum_{i=1}^{m} f\left(x^{(i)}\right)\chi_S\left(x^{(i)}\right)$
end
return Hypothesis $h(x) = \operatorname{sgn}\left(\sum_{S\in\mathscr{F}} \widetilde{f}(S)\chi_S(x)\right)$
```

Algorithm 1: Kushilevitz-Mansour-Algorithm

Equipped with this statement we are able to state the Kushilevitz-Mansour-Algorithm (KM-Algorithm, see Algorithm 1) and prove its correctness.

Theorem 3.16. *(Kushilevitz-Mansour-Algorithm) Let $f : \{-1,1\}^n \to \{-1,1\}$ have its Fourier spectrum $\varepsilon/2$-concentrated on collection $\mathscr{F}$. Given $m \geq 4|\mathscr{F}|\ln\left(2|\mathscr{F}|/\delta\right)/\varepsilon$ random examples drawn independently from the uniform distribution the algorithm outputs a hypothesis $h : \{-1,1\}^n \to \{-1,1\}$ which with probability $1-\delta$ satisfies* $\operatorname{dist}(f,h) \leq \varepsilon$.

Proof. We first assume that the algorithm produces approximations $\widetilde{f}(S)$ of the Fourier coefficients $\widehat{f}(S)$ with $\left|\widetilde{f}(S) - \widehat{f}(S)\right| < \lambda$. This can be expressed by Lemma 2.4 since each $f(\boldsymbol{x}^{(i)})\chi_S(\boldsymbol{x}^{(i)}) \in \{-1,1\}$ is a random variable with expected value $\mathbf{E}\left[f(\boldsymbol{x}^{(i)})\chi_S(\boldsymbol{x}^{(i)})\right] = \widehat{f}(S)$. Hence we have

$$\mathbf{Pr}\left[\left|\widetilde{f}(S) - \widehat{f}(S)\right| \geq \lambda\right] \leq 2e^{-\lambda^2 m/2}. \tag{3.3.1}$$

We define $g : \{-1,1\} \to \mathbb{R}$ with

$$g(x) = \sum_{S\in\mathscr{F}} \widetilde{f}(S)\chi_S(x),$$

and $h(x) = \operatorname{sgn}(g(x))$ will be our output.

Considering the case in which all approximations are actually within accuracy λ, we look at the error of g concerning f:

$$\begin{aligned}
\mathop{\mathbf{E}}_{\boldsymbol{x}}\left[(f(\boldsymbol{x}) - g(\boldsymbol{x}))^2\right] &= \sum_{S\subseteq[n]} \widehat{f-g}(S)^2 && \text{(Theorem 2.5)}\\
&= \sum_{S\subseteq[n]} \left(\widehat{f}(S) - \widetilde{f}(S)\right)^2 \\
&= \sum_{S\notin\mathscr{F}} \widehat{f}(S)^2 + \sum_{S\in\mathscr{F}} \left(\widehat{f}(S) - \widetilde{f}(S)\right)^2 \\
&\leq \varepsilon/2 + \sum_{S\in\mathscr{F}} \lambda^2 && \text{(concentration, accuracy)}\\
&= \varepsilon/2 + |\mathscr{F}|\lambda^2
\end{aligned}$$

Since we need $\operatorname{dist}(f,h) \leq \varepsilon$, by Proposition 3.15 we have

$$|\mathscr{F}|\lambda^2 \leq \varepsilon/2 \Leftrightarrow \lambda \leq \sqrt{\frac{\varepsilon}{2|\mathscr{F}|}}.$$

This accuracy has to hold with probability $1-\delta$ for all approximated coefficients. We hence require

$$\mathbf{Pr}\left[\bigvee_{S\in\mathscr{F}} \left|\widehat{f}(S) - \widetilde{f}(S)\right| > \lambda\right] \leq \delta.$$

By the union bound it suffices if $\mathbf{Pr}\left[\left|\widehat{f}(S) - \widetilde{f}(S)\right| > \lambda\right] \leq \delta/|\mathscr{F}|$ for all $S \in \mathscr{F}$. From (3.3.1) we thus obtain

$$\begin{aligned} 2e^{-\lambda^2 m/2} &\leq \frac{\delta}{|\mathscr{F}|} \\ \frac{-\lambda^2 m}{2} &\leq \ln\left(\frac{\delta}{2|\mathscr{F}|}\right) \\ \frac{\varepsilon}{2|\mathscr{F}|} m &\geq 2\ln\left(\frac{2|\mathscr{F}|}{\delta}\right) \\ m &\geq \frac{4|\mathscr{F}|}{\varepsilon}\ln\left(\frac{2|\mathscr{F}|}{\delta}\right). \end{aligned}$$

□

The runtime of the KM-Algorithm is $\mathcal{O}\left(|\mathscr{F}|^2 \ln\left(|\mathscr{F}|/\delta\right)/\varepsilon\right)$ since we need to compute a sum with m addends for all $|\mathscr{F}|$ coefficients we want to approximate.

A special case of the KM-Algorithm is the Low-Degree-Algorithm. The algorithm takes a parameter $k \geq 0$ and computes the KM-Algorithm with $\mathscr{F} = \{S \subseteq [n] \mid |S| \leq k\}$. This is useful because we can often make statements about the weight concentration of a function concerning the degree of the Fourier coefficients. The amount of sets up to degree k is bounded by $\mathcal{O}\left(n^k\right)$. Hence the runtime of the Low-Degree-Algorithm is $\mathcal{O}\left(n^{2k}\ln\left(n^k/\delta\right)/\varepsilon\right)$. To use the results of the influence we show how the total influence and ε-concentration are related.

Proposition 3.17. *Let $f : \{-1,1\}^n \to \{-1,1\}$ and $\varepsilon > 0$. Then the Fourier spectrum of f is ε-concentrated up to degree $\mathbf{I}[f]/\varepsilon$.*

Proof. We have

$$\sum_{\substack{S\subseteq[n]\\|S|>k}} \widehat{f}(S)^2 \leq \sum_{\substack{S\subseteq[n]\\|S|>k}} \frac{|S|}{k}\widehat{f}(S)^2,$$

because $|S|/k > 1$. We can add all remaining addends with $|S| \leq k$ without breaking the inequality. Hence we get

$$\begin{aligned}\sum_{\substack{S \subseteq [n] \\ |S| > k}} \widehat{f}(S)^2 &\leq \sum_{S \subseteq [n]} \frac{|S|}{k} \widehat{f}(S)^2 \\ &= \frac{\mathbf{I}[f]}{k}. && \text{(Theorem 3.8)}\end{aligned}$$

By substituting $k = \mathbf{I}[f]/\varepsilon$ we obtain

$$\sum_{\substack{S \subseteq [n] \\ |S| > \mathbf{I}[f]/\varepsilon}} \widehat{f}(S)^2 \leq \varepsilon.$$

$\square$

Combining this fact with the Low-Degree-Algorithm and the results of Theorem 3.10 we can make a statement about the PAC learnability of Arbiter PUFs.

Corollary 3.18. *Let $\mathscr{C} = \{f : \{-1,1\}^n \to \{-1,1\} \mid f \text{ is an ATF}\}$, then $\mathscr{C}$ is learnable from random examples with error ε in time $n^{\mathcal{O}(n/\varepsilon)}$.*

Sadly this result is not too impressive since any function $f : \{-1,1\}^n \to \{-1,1\}$ can be learned with error $\varepsilon = 0$ in time $\widetilde{\mathcal{O}}(2^n)$. There is however an interesting aspect on this result. Gotsman and Linial have shown an upper bound for the total influence of LTFs which is $\mathbf{I}[f_{\text{LTF}}] \leq \sqrt{n}$ [GL94]. This implies a learnability in time $n^{\mathcal{O}(\sqrt{n}/\varepsilon)}$ which is still not very good but as we showed in Theorem 2.14 an ATF can be expressed as an LTF by applying a bijection on the input vector. We can see that by applying this transformation we can improve the runtime needed for learning. As we saw in Section 2.4 this method is also used in the current non-PAC state-of-the-art attacks.

3.3.2 *PAC Learning of Combined Arbiter PUFs*

Next we will make the step from a single, to the combination of multiple Arbiter PUFs. Klivans et. al., building on the results of Peres [Per04], were able to deduce a new result on learnability of combined LTFs [KOS02].

Theorem 3.19. *Let $g : \{-1,1\}^k \to \{-1,1\}$ be any Boolean function on k bits and $f_1, \ldots, f_k : \{-1,1\}^n \to \{-1,1\}$ be distinct LTFs. Then $\mathscr{C} = \{h(x) = g(f_1(x), \ldots, f_k(x))\}$ is learnable from uniformly drawn random examples with error ε in time $n^{\mathcal{O}(k^2/\varepsilon^2)}$.*

We can for instance apply this to the XOR Arbiter PUF using the model presented in (2.3.1). In fact it does not matter which function is used to combine the PUFs, the resulting construct is always PAC learnable in time poly(n). An important note is that this result is yet practically not significant since the constants in the exponent are too big for an attack to be efficient. However it shows a theoretical upper bound for the complexity of learning a construct of combined Arbiter PUFs. Additionally, another aspect to look at are the exact prerequisites to use this theorem for an attack. The attack only works if all of the combined LTFs get the same input. It cannot be applied if the PUF is constructed in a way that the PUF input is transformed such that each LTF obtains a different individual input. This is for instance the case in the Lightweight Secure PUF [MKP08]. However if there is a transformation that is the same for all Arbiter PUFs, the attacker can compute this transformation to adjust the CRPs for learning to hold the actual inputs of the LTFs. Hence, in this case the attack would still be feasible.

4 CONCLUSION

In this thesis we looked at Arbiter PUFs as well as XOR Arbiter PUFs from the perspective of Boolean functions. For this purpose we presented how the Arbiter PUF can be described by a mathematical model. We introduced the notion of influence and its relation to the Fourier expansion before we applied it to the model of the PUFs. Doing so, we determined the expected value of the influence for a random Arbiter PUF instance and extended the results to the total influence.

We also approximated the distribution of the influence for Arbiter PUFs and empirically estimated the alteration of its variance with increasing input length and for different input bit positions. Using a recursive approach we extended the results to the XOR Arbiter PUF. The same technique was also employed to determine a closed form for the stability of an XOR Arbiter PUF. We utilized the results of the influence to deduce statements about the PAC learnability of Arbiter PUFs using the KM-Algorithm. The results, however, are of theoretical value, since practical results exceed in runtime and efficiency. An interesting aspect is that we can reduce the theoretical runtime for learning, by transforming an ATF to an LTF. This fact is used, for instance, in Rührmair's LR attack in a way that the CRPs are transformed to learn LTFs rather than ATFs.

Afterward, we deduced from Klivans et. al. that a combination of k Arbiter PUFs of length n can be learned with error ε in time $n^{\mathcal{O}\left(k^2/\varepsilon^2\right)}$. This shows that changing only the function used for the combination, will not alter the fact that the resulting PUF is learnable in time $\operatorname{poly}(n)$. Although a practical application is not realistic at the moment, since other attacks are performing significantly better, the results are evidence that changing the combiner function will not render the practical attacks inefficient.

In future work it will be interesting to find a PAC learning algorithm for XOR Arbiter PUFs that can compete with the known state-of-the-art attacks. Further analysis of the Arbiter PUF's Fourier spectrum may be a way to find characteristic structures that can be utilized for the creation of such an algorithm. From this knowledge it could be possible to create a construct that can better withstand known attacks. Another imaginable goal is to understand the exact prerequisites for a successful ML attack to find potential countermeasures. Finally, since delay based PUFs can be efficiently attacked with photonic emission analysis, it is desirable to develop alternative

PUF concepts, for instance, based on established demands on cryptographic Boolean functions.

BIBLIOGRAPHY

[Bec15] Georg T Becker. “The Gap Between Promise and Reality: On the Insecurity of XOR Arbiter PUFs.” en. In: *Cryptographic Hardware and Embedded Systems – CHES 2015*. Vol. 9293. Springer, Berlin, Heidelberg, 13 9 2015, pp. 535–555 (cit. on p. 20).

[DGM05] Deepak Kumar Dalai, Kishan Chand Gupta, and Subhamoy Maitra. “Cryptographically Significant Boolean Functions: Construction and Analysis in Terms of Algebraic Immunity.” In: *FSE*. Vol. 3557. 2005, pp. 98–111 (cit. on p. 4).

[DV13] Jeroen Delvaux and Ingrid Verbauwhede. “Side channel modeling attacks on 65nm arbiter PUFs exploiting CMOS device noise.” In: *2013 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*. June 2013, pp. 137–142 (cit. on p. 13).

[Dev09] Srini Devadas. “Physical Unclonable Functions and Secure Processors.” en. In: *Cryptographic Hardware and Embedded Systems - CHES 2009*. Vol. 5747. Springer, Berlin, Heidelberg, 2009, pp. 65–65 (cit. on p. 14).

[Eis+08] Thomas Eisenbarth, Timo Kasper, Amir Moradi, Christof Paar, Mahmoud Salmasizadeh, and Mohammad T Manzuri Shalmani. “On the Power of Power Analysis in the Real World: A Complete Break of the KeeLoq Code Hopping Scheme.” In: *Advances in Cryptology – CRYPTO 2008*. Ed. by David Wagner. Vol. 5157. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 203–220 (cit. on p. 1).

[Gas+02] Blaise Gassend, Dwaine Clarke, Marten van Dijk, and Srinivas Devadas. “Silicon physical random functions.” In: *Proceedings of the 9th ACM conference on Computer and communications security*. ACM, 18 11 2002, pp. 148–160 (cit. on pp. 3, 9).

[GL94] Craig Gotsman and Nathan Linial. “Spectral properties of threshold functions.” en. In: *Combinatorica* 14.1 (Jan. 1994), pp. 35–50 (cit. on p. 38).

[Gua+07] Jorge Guajardo, Sandeep S Kumar, Geert-Jan Schrijen, and Pim Tuyls. “FPGA Intrinsic PUFs and Their Use for IP Protection.” In: *Cryptographic Hardware and Embedded Systems - CHES 2007*. Ed. by Pascal Paillier and Ingrid Verbauwhede. Vol. 4727. Lecture Notes in Computer Science.

Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 63–80 (cit. on p. 3).

[Guo+16] Qingli Guo, Jing Ye, Yue Gong, Yu Hu, and Xiaowei Li. "Efficient Attack on Non-linear Current Mirror PUF with Genetic Algorithm." In: *2016 IEEE 25th Asian Test Symposium (ATS)*. IEEE, Nov. 2016, pp. 49–54 (cit. on p. 3).

[HS08] Ghaith Hammouri and Berk Sunar. "PUF-HB: A Tamper-Resilient HB Based Authentication Protocol." en. In: *Applied Cryptography and Network Security*. Vol. 5037 LNCS. Springer, Berlin, Heidelberg, Mar. 2008, pp. 346–365 (cit. on p. 26).

[Hoe63] Wassily Hoeffding. "Probability Inequalities for Sums of Bounded Random Variables." In: *J. Am. Stat. Assoc.* 58.301 (Jan. 1963), pp. 13–30 (cit. on p. 7).

[KSP10] Timo Kasper, Michael Silbermann, and Christof Paar. "All You Can Eat or Breaking a Real-World Contactless Payment System." In: *Financial Cryptography and Data Security*. Ed. by Radu Sion. Vol. 6052. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 343–350 (cit. on p. 1).

[KOS02] Adam R Klivans, Ryan O'Donnell, and Rocco A Servedio. "Learning intersections and thresholds of halfspaces." In: *The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings.* Nov. 2002, pp. 177–186 (cit. on p. 38).

[KB14] Raghavan Kumar and Wayne Burleson. "On design of a highly secure PUF based on non-linear current mirrors." In: *2014 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*. IEEE, May 2014, pp. 38–43 (cit. on p. 3).

[Lim04] Daihyun Lim. "Extracting Secret Keys from Integrated Circuits." MA thesis. Massachusetts Institute Technology, May 2004 (cit. on pp. 3, 14).

[MKP08] Mehrdad Majzoobi, Farinaz Koushanfar, and Miodrag Potkonjak. "Lightweight secure PUFs." In: *2008 IEEE/ACM International Conference on Computer-Aided Design*. Vol. 1. IEEE, Nov. 2008, pp. 670–673 (cit. on p. 39).

[Man94] Yishay Mansour. "Learning Boolean Functions via the Fourier Transform." en. In: *Theoretical Advances in Neural Computation and Learning*. Ed. by Vwani Roychowdhury, Kai-Yeung Siu, and Alon Orlitsky. Springer US, 1994, pp. 391–424 (cit. on p. 35).

[OD014] Ryan O'Donnell. *Analysis of Boolean Functions*. en. New York, NY, USA: Cambridge University Press, May 2014 (cit. on p. 5).

[Pap+02] Ravikanth Pappu, Ben Recht, Jason Taylor, and Neil Gershenfeld. "Physical one-way functions." en. In: *Science* 297.5589 (20 9 2002), pp. 2026–2030 (cit. on p. 2).

[Pen46] L S Penrose. "The Elementary Statistics of Majority Voting." In: *J. R. Stat. Soc.* 109.1 (1946), pp. 53–57 (cit. on p. 4).

[Per04] Yuval Peres. "Noise Stability of Weighted Majority." In: *arXiv [math]* January 2005 (Dec. 2004) (cit. on p. 38).

[Rüh+13] Ulrich Rührmair, Jan Sölter, Frank Sehnke, Xiaolin Xu, Ahmed Mahmoud, Vera Stoyanova, Gideon Dror, Jürgen Schmidhuber, Wayne Burleson, and Srinivas Devadas. "PUF Modeling Attacks on Simulated and Silicon Data." In: *IEEE Trans. Inf. Forensics Secur.* 8.11 (Nov. 2013), pp. 1876–1891 (cit. on p. 17).

[Sko05] Sergei P Skorobogatov. *Semi-invasive attacks – A new approach to hardware security analysis*. 2005 (cit. on p. 1).

[SD07] G Edward Suh and Srinivas Devadas. "Physical Unclonable Functions for Device Authentication and Secret Key Generation." In: *2007 44th ACM/IEEE Design Automation Conference*. IEEE, June 2007, pp. 9–14 (cit. on pp. 3, 14).

[Taj+17] Shahin Tajik, Enrico Dietz, Sven Frohmann, Helmar Dittrich, Dmitry Nedospasov, Clemens Helfmeier, Jean-Pierre Seifert, Christian Boit, and Heinz-Wilhelm Hübers. "Photonic Side-Channel Analysis of Arbiter PUFs." en. In: *J. Cryptology* 30.2 (Jan. 2017), pp. 550–571 (cit. on p. 21).

[Val84] Leslie G Valiant. "A theory of the learnable." In: *Commun. ACM* 27.11 (1984), pp. 1134–1142 (cit. on p. 34).

[VK15] Arunkumar Vijayakumar and Sandip Kundu. "A novel modeling attack resistant PUF design based on non-linear voltage transfer characteristics." In: *2015 Design, Automation Test in Europe Conference Exhibition (DATE)*. Mar. 2015, pp. 653–658 (cit. on p. 3).

[Wal23] J L Walsh. "A Closed Set of Normal Orthogonal Functions." In: *Amer. J. Math.* 45.1 (1923), pp. 5–24 (cit. on p. 4).

[WT86] A F Webster and Stafford E Tavares. "On the design of S-boxes." In: *Advances in Cryptology-CRYPTO'85: Proceedings*. 1986, p. 523 (cit. on p. 34).

DECLARATION

I hereby certify that this thesis has been composed by me and is based on my own work, unless stated otherwise. No other person's work has been used without due acknowledgments in this thesis. All references and verbatim extracts have been quoted, and all sources of information, including graphs and data sets, have been specifically acknowledged.

Berlin, 24th July 2017

Benjamin Zengin